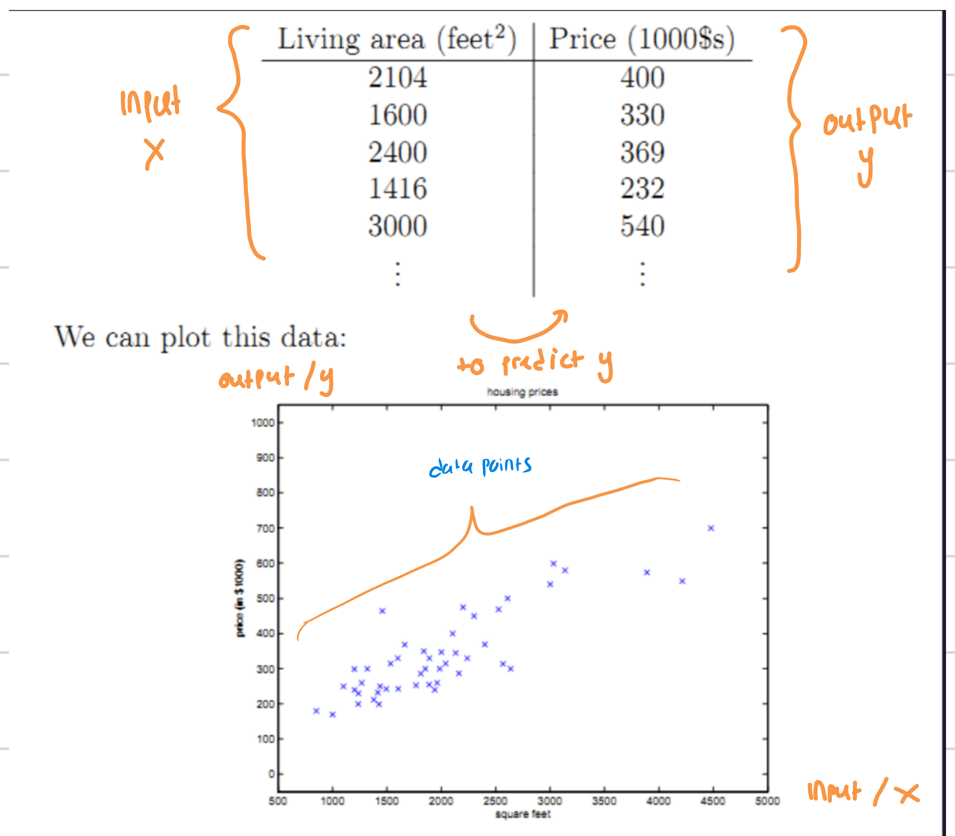
Linear Regression

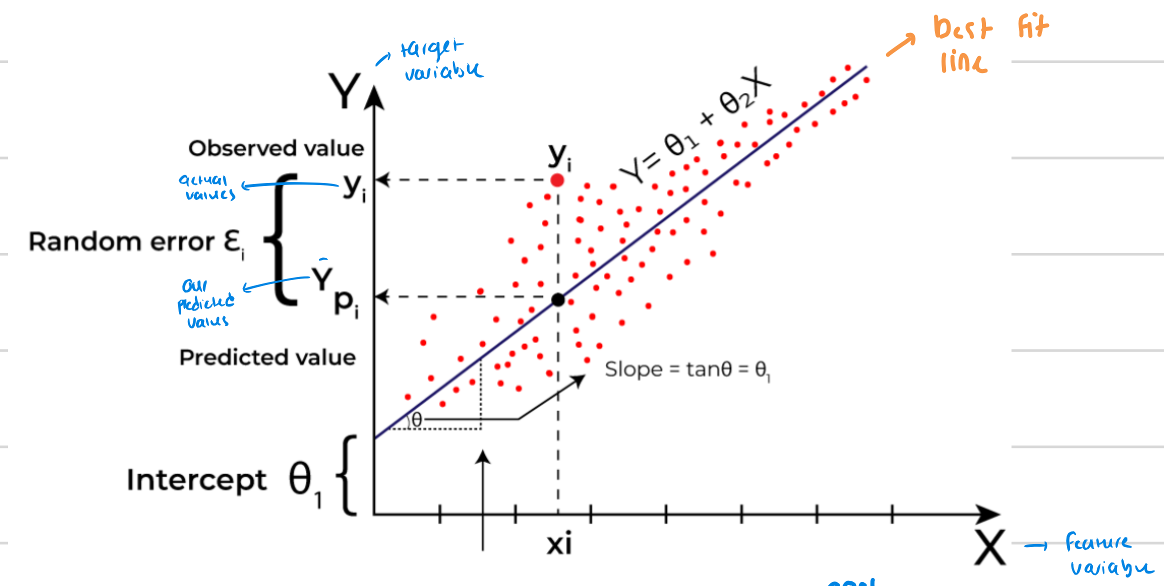
Algorithm that provides a linear relationship between an independent variable (input / features / x) and dependent variable (output / target / y) to predict (prediction model) outcome of future events



Living area (feet^2)	Price (1000$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮

We can plot this data:

Goal => find the best fit line equation that can predict the values based on independent var.



Best fit line = Make the $\underbrace{\text{Predicted Target Value, } \hat{Y} - \text{Actual Target Value, } Y}_{\text{cost function}}$ to the most minimum.

Cost Function = $\underset{\text{predicted}}{\hat{Y}} - \underset{\text{actual}}{Y}$ => MSE (mean square error) cost function is used

↳ calculates the ave. squared errors between $\underset{\text{predict}}{\hat{Y}} - \underset{\text{actual}}{Y}$

↳ to determine optimal value for b & coeff. of X_i => giving best fit line

$$\text{Cost Function } (J) = \underset{\text{num of data points}}{\frac{1}{n}} \sum_{i=1}^{n} \underbrace{(\hat{y}_i - y_i)^2}_{\text{sum of cost function squared}} = \underset{\text{ave. cost function squared}}{MSE}$$

$$\underset{\text{output/target}}{y} = \underset{\text{regression coeff}}{m}\,\underset{\text{input / feature}}{x} + \underset{\text{bias term}}{b}$$ => Simple Linear Regression

$$y = \overbrace{m_1 x_1}^{\text{feature 1}} + \overbrace{m_2 x_2}^{\text{feature 2}} + \ldots + b$$ => Multiple Linear Regression

Ways to Minimize Cost Function (MSE)

① Least Mean Square

↳ $J(\theta) = \frac{1}{2} \sum_{i=1}^{m} [\underbrace{h_\theta(x^i)}_{\text{predicted target}} - \underbrace{y^i}_{\text{actual target}}]^2$ (m = num of examples; cost function)

1. Choose initial guess for θ ie. predicted target
2. Check the cost function, $J(\theta)$
3. if $J(\theta) < J(\theta)_{\text{previous}}$ => change θ
4. repeat until $J(\theta)$ is not lower than previous

② Batch Gradient Descent

↳ Modify the models parameters to minimize cost function

1. Choose initial guess for θ
2. Calculate the cost function (J) over the entire dataset
3. differentiate the cost function (J) w/ respect to the model parameters
4. update the model parameters $\underset{\text{model param}}{\theta} := \underset{\text{model param}}{\theta} - \underset{\text{learning rate}}{\alpha} \underset{\text{diff. of cost function}}{\nabla J(\theta)}$
5. Repeat until cost function reach the lowest (convergence)

③ incremental gradient descent

↳ like Batch Gradient Descent but update the parameter after each training example not the whole dataset

1. Choose initial guess for θ
2. calculate the cost function for the training example
3. update the model parameter
4. Repeat w/ other training example until reach convergence

④ Locally weighted Regression

↳ Non-parametric algorithm that fits multiple regression models to localized subsets

1. For each query_point, assign weight to all training points based on their distance to query_point. ↓ distance, ↑ weights
2. fit a linear regression model to the weighted training data
3. use the fitted model to make predictions for query_point
4. Repeat for each query_point

⑤ Logistic Regression

↳ algorithm for categorical output Y

1. Predict the probability of output Y being 1 given inputs X using sigmoid ∫ function $\sigma(z) = \frac{1}{1+e^{-z}}$
2. Use Incremental Gradient Descent to maximize the probability