Supervised learning

- Data Preparation and Splitting
  - Train, validation, test approach
  - K-fold Cross validation approach
- Regression
  - Bias-variance tradeoff
  - Relationship with model complexity
  - Underfitting and Overfitting
  - Linear regression
  - Evaluation metrics

Splitting the data

Original

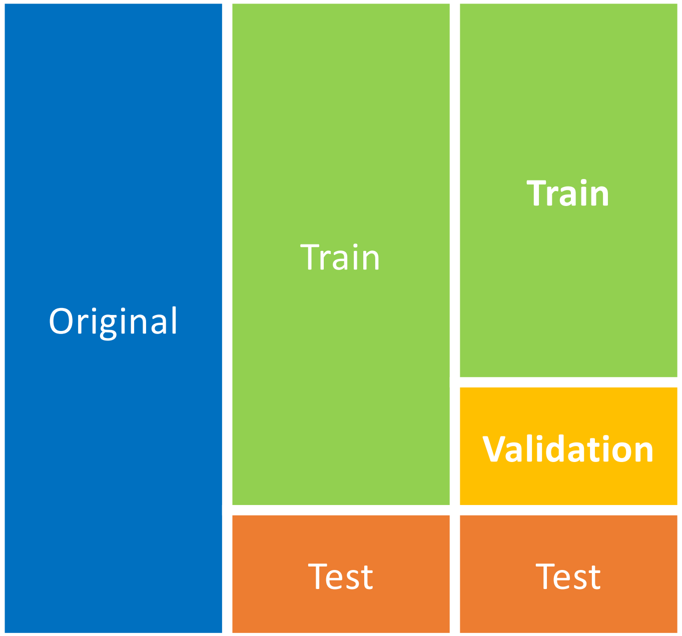
- Ensure that the instances are representative and relevant to the problem
- Ensure collaboration in data preprocessing
- *It's a good practice to shuffle the data before the split to avoid bias in the resulting sets*

Train, test, and validation split approach

In most supervised machine learning tasks, best practice recommends to split your data into three independent sets:

- Training set
- Validation set
- Testing set (It must not be touched until the end)

Train, test, and validation split approach



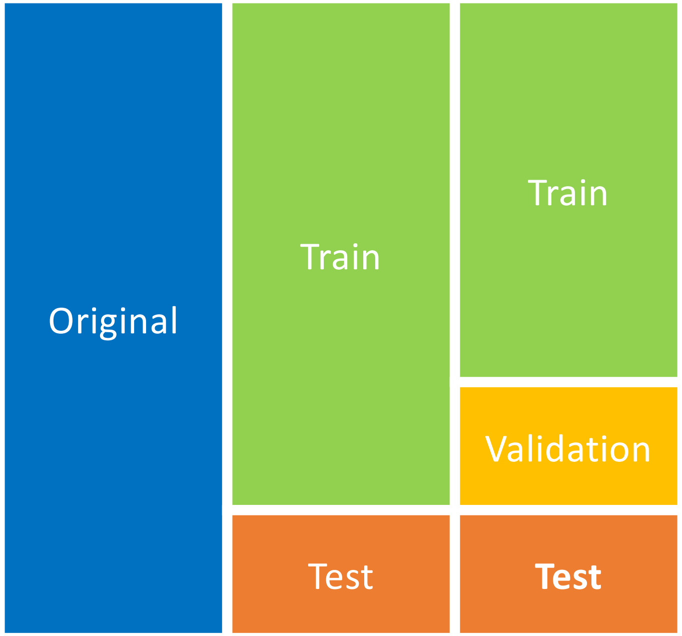
- Used to train the model
- Find patterns
- Class examples
- Build models

- Used for unbiased evaluation of the model

Iterative learning process

Train -> tune -> validate the model

Train, test, and validation split approach

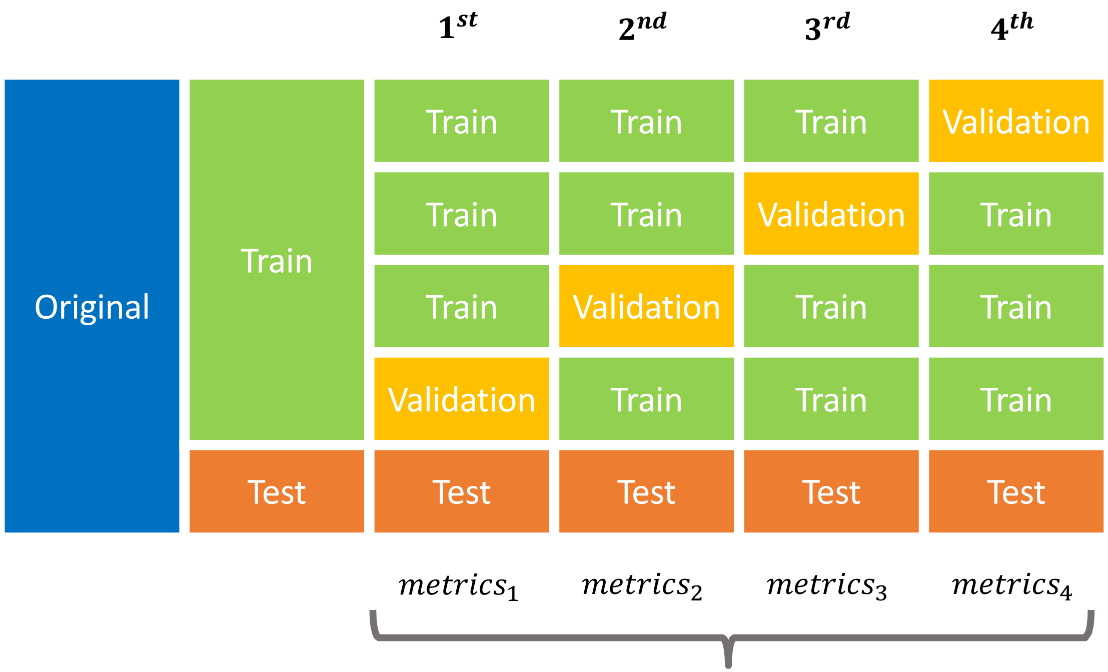


- It is not available to the model for learning
- It is only used to ensure that the model generalizes well on new "unseen" data
- It is used for final evaluation of the model

K-Fold Cross-Validation approach

- This strategy iteratively use different portions (folds) of our data to train and validate our model
- Useful when data is limited and we want more robust estimates of model performance

K-Fold Cross-Validation approach (e.g., k=4)

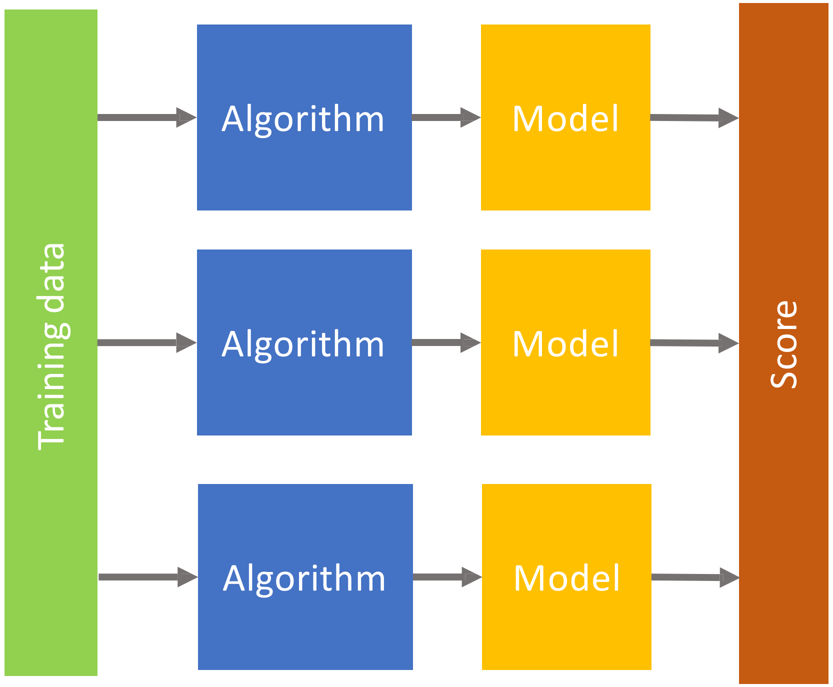


The final **test** is evaluated by average performance on the test set metrics across all the folds

1. Split the **train** into k independent folds
2. Repeat the following k times:
   1. Set aside k^{th} fold (hold-out set) of the data for **validation**
   2. Train the model on the remaining $k-1$ folds
   3. Test the model on the **validation**
3. At the end, average or combine the model performance metrics

*Test data remains unchanged

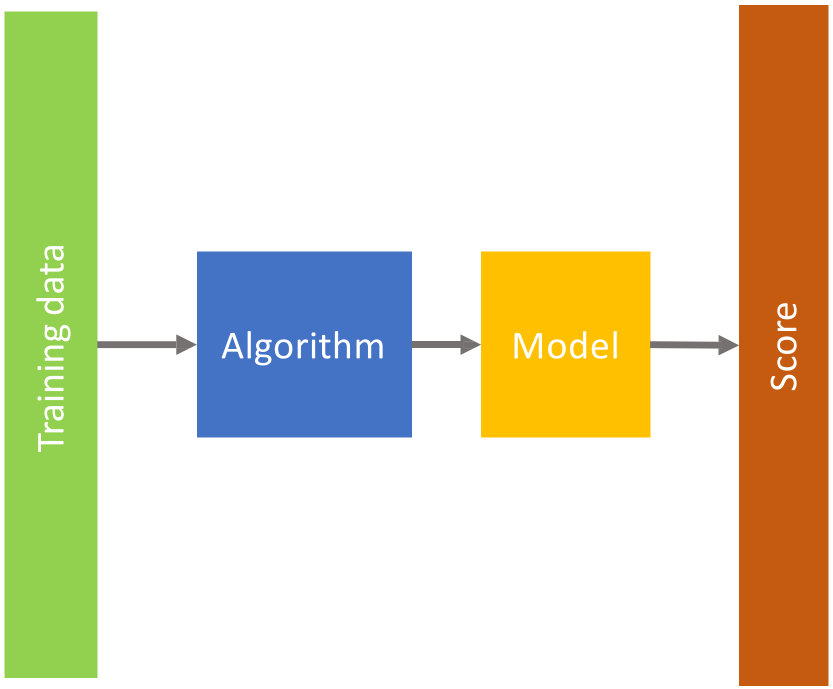
Process preview



Choice a ML algorithm

- Suitable for desired label
- Suitable for learning type
- Suitable for data size
- Suitable given available info
- Suitable given practical issues
  - Production requirements
  - Engineering effort
  - Expertise required

Process preview



Choice a ML algorithm

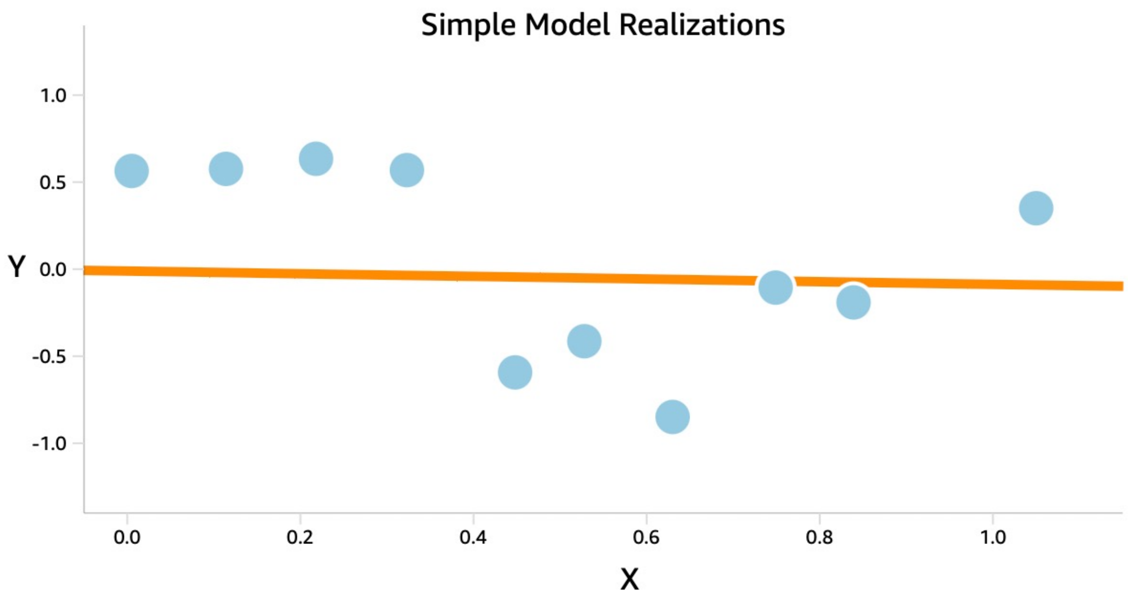
- Suitable for desired label
- Suitable for learning type
- Suitable for data size
- Suitable given available info
- Suitable given practical issues
  - Production requirements
  - Engineering effort
  - Expertise required

The bias-variance tradeoff

- Simple model

A simple model

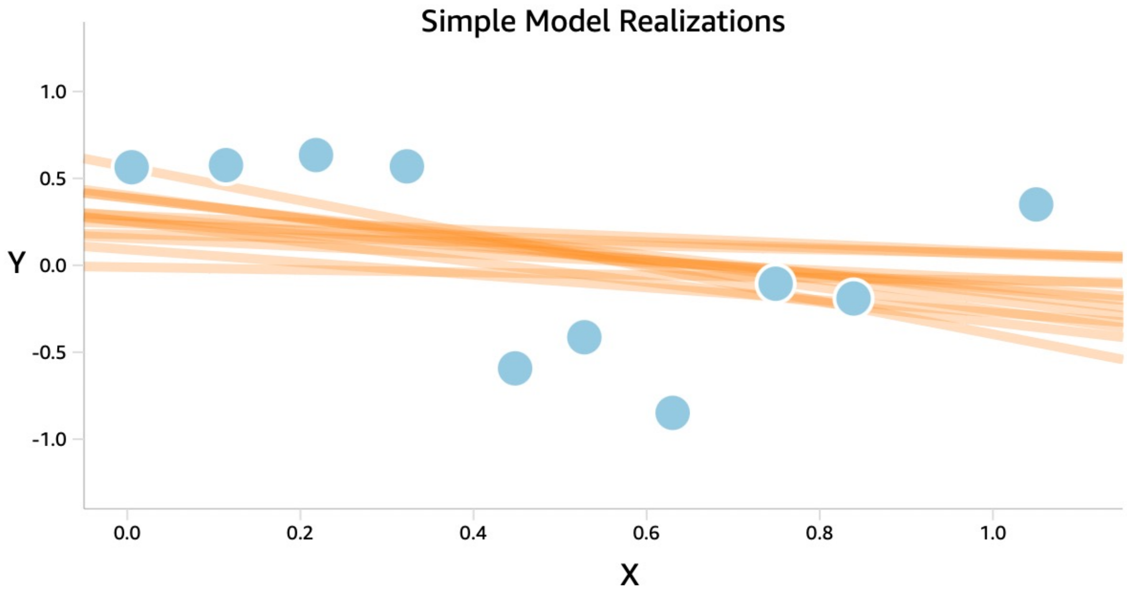
No matter how we try to fit this line, *the straight line does not have the flexibility to accurately capture the relationship of the data*



The model is to simple to capture the structure of the data

A simple model

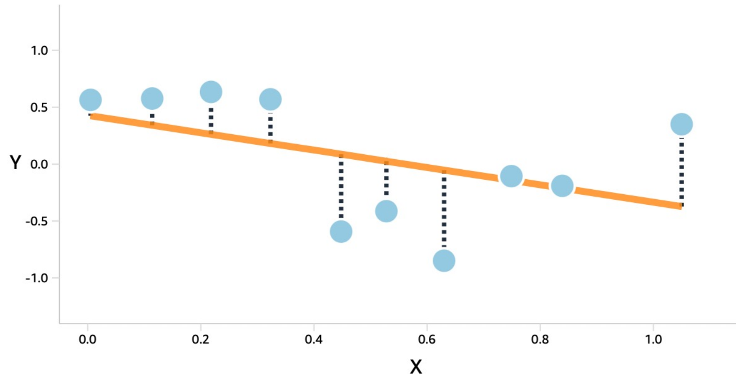
No matter how we try to fit this line, *the straight line does not have the flexibility to accurately capture the relationship of the data*

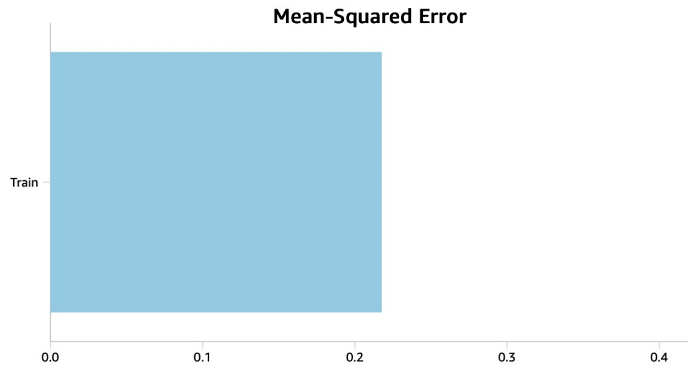


The model is to simple to capture the structure of the data

A simple model

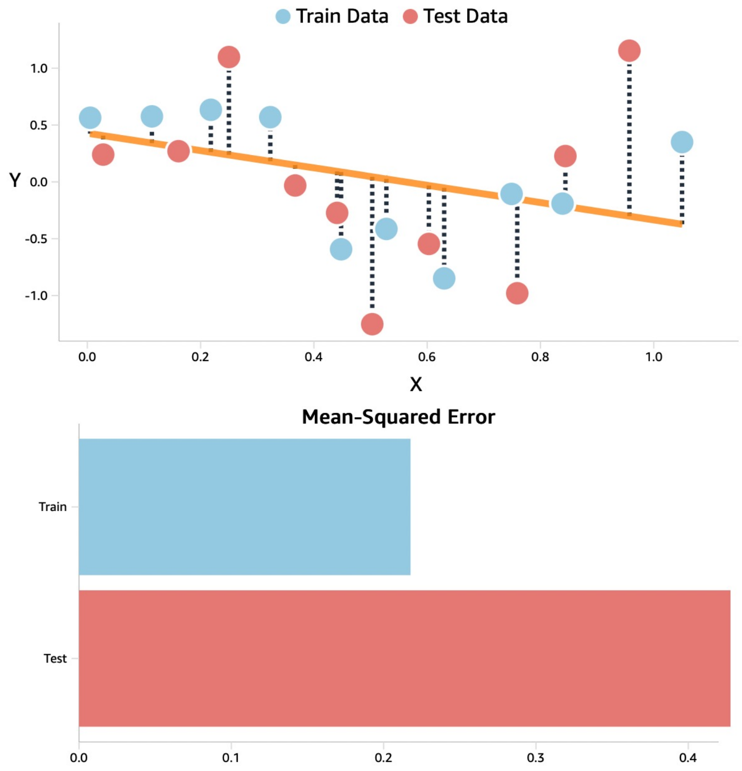
- No matter how we try to fit this line, *the straight line does not have the flexibility to accurately capture the relationship of the data*
- Error in the **training data** is not zero





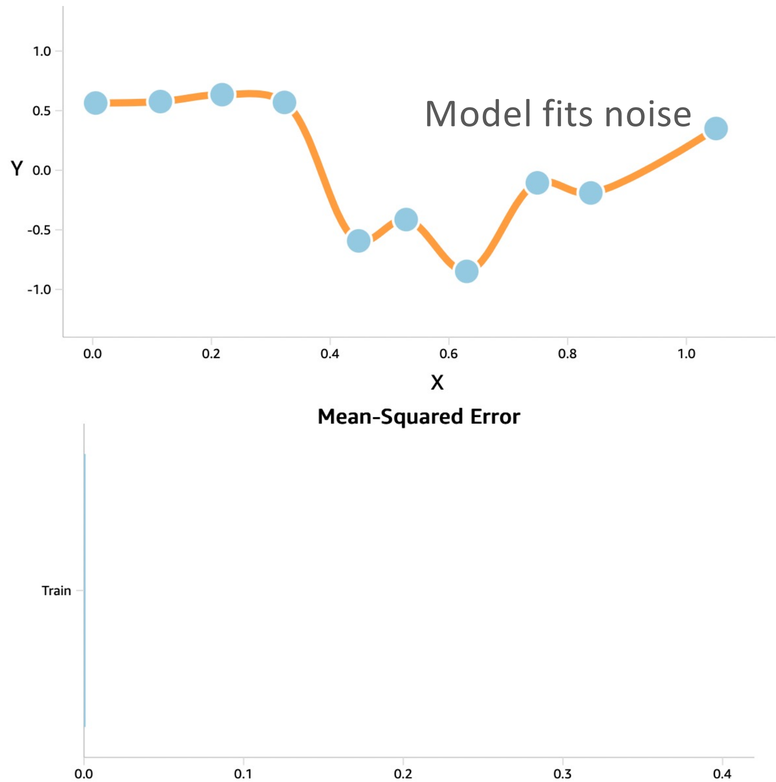
Underfitting: low complexity model

- **High error** on train data and test data
- The model is so simple that it fails to adequately capture the relationships in the data
- The high error on test is a direct result of the lack of complexity of the model



A complex model

- A more complex model predicts every point in the training data perfectly
- Super flexible model
- Error in the **train data** is zero



Overfitting: high complex model

- **Low error** on train data and **high** on test data
- The model is so specific to the data on which it was training that is is no longer applicable to different datasets
- The model is not generalizable to other datasets (e.g., test data) ⚠️

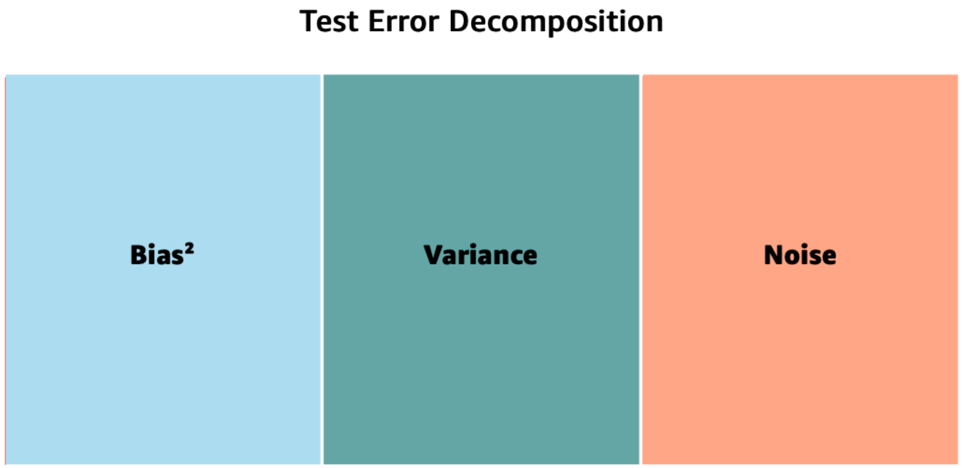
Test error decomposition

Mean Squared Error (MSE) can be decomposed into three components:

- $Bias^2$: Error due to wrong assumptions
- $Variance$: Error due to sensitivity to data
- $Irreducible\ error$: Error due to noise in data

$$Error = Bias^2 + Variance + Noise$$

$$Error = (\mathrm{E}[\hat{f}(x)] - f(x))^2 + \mathrm{E}\left[\left(\hat{f}(x) - \mathrm{E}[\hat{f}(x)]\right)^2\right] + Noise$$



We can make use of the relationship between both bias and variance to obtain better predictions.

Bias and variance

Bias

- It measure how far the **average** model prediction is from the **actual** outcome
- High bias usually means the model is too **simple** to capture the underlying patterns (underfitting)

Variance

- It measure how much the model's prediction **change** when trained on different samples of the data
- High variance often means the model is too **complex** and sensitive to noise (overfitting)

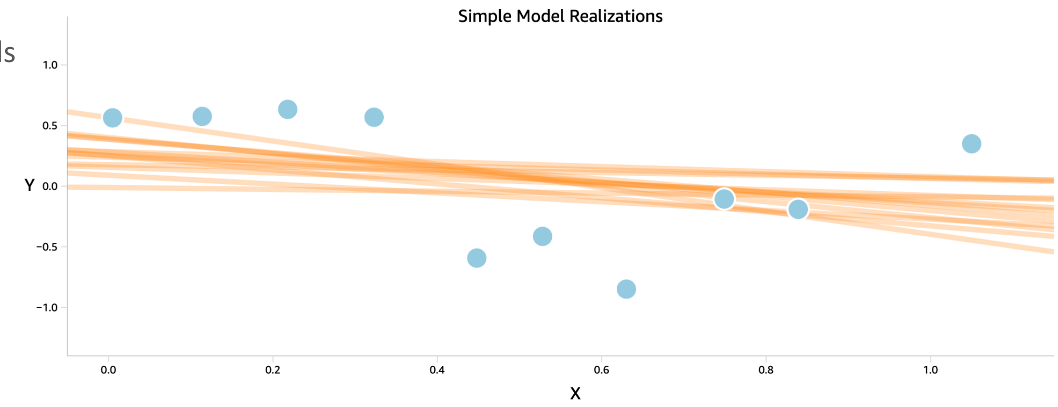
$$Bias(x) = \mathrm{E}\left[\hat{f}(x)\right] - f(x)$$

$$Variance(x) = \mathrm{E}[(\hat{f}(x) - \mathrm{E}\left[\hat{f}(x)\right])^2]$$

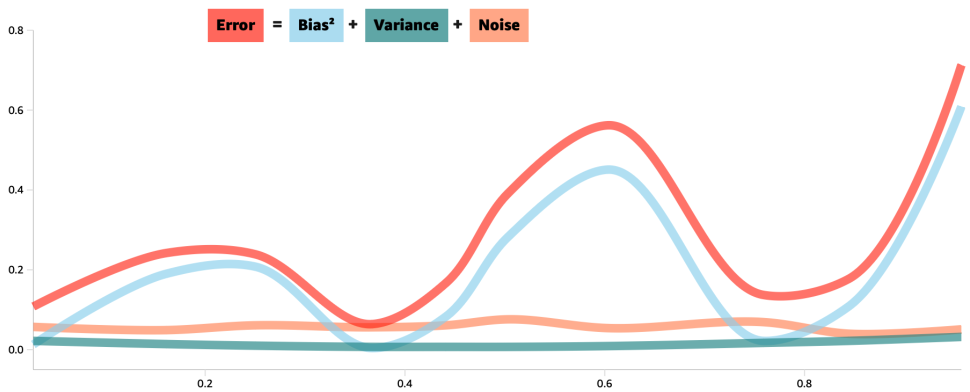
$\hat{f}(x)$: the prediction made by a trained model
$\mathrm{E}\left[\hat{f}(x)\right]$: the expected prediction from many models trained on different random training datasets
$f(x)$: the true value (true function or ground truth)

Bias and variance

Different models



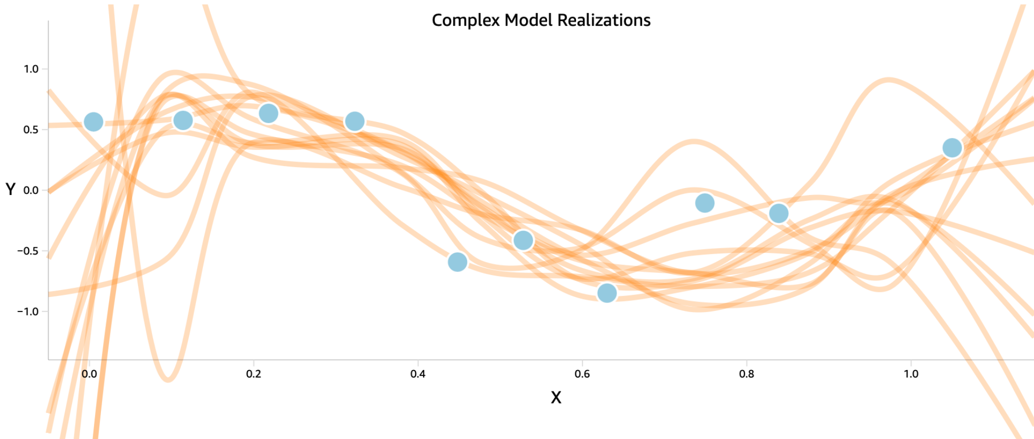
Error decomposition
(for each point of data)



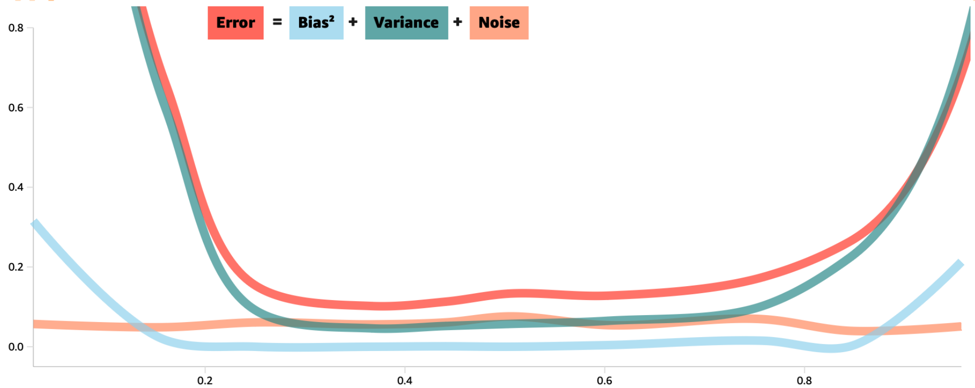
For underfit models (low-complexity), the majority of the error comes from **bias**

Bias and variance

Different models



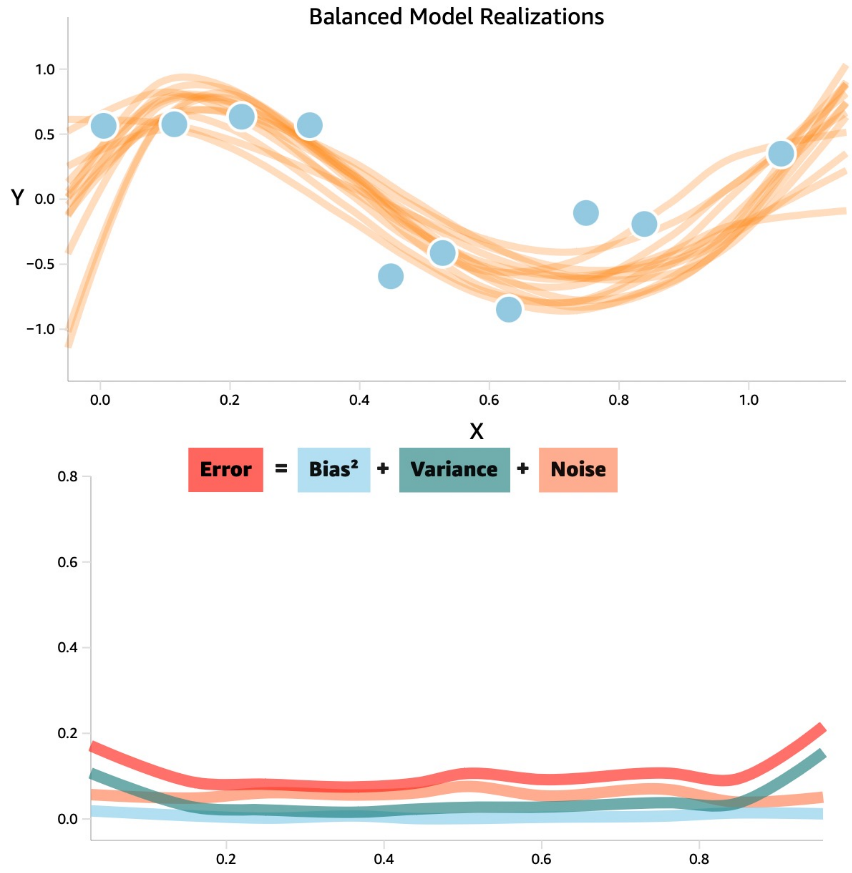
Error decomposition
(for each point of data)



Overfit models (high-complexity) show a lot more error from **variance** than from bias. It's easy to imagine that any unseen data points will be predicted with high error

Find a balance

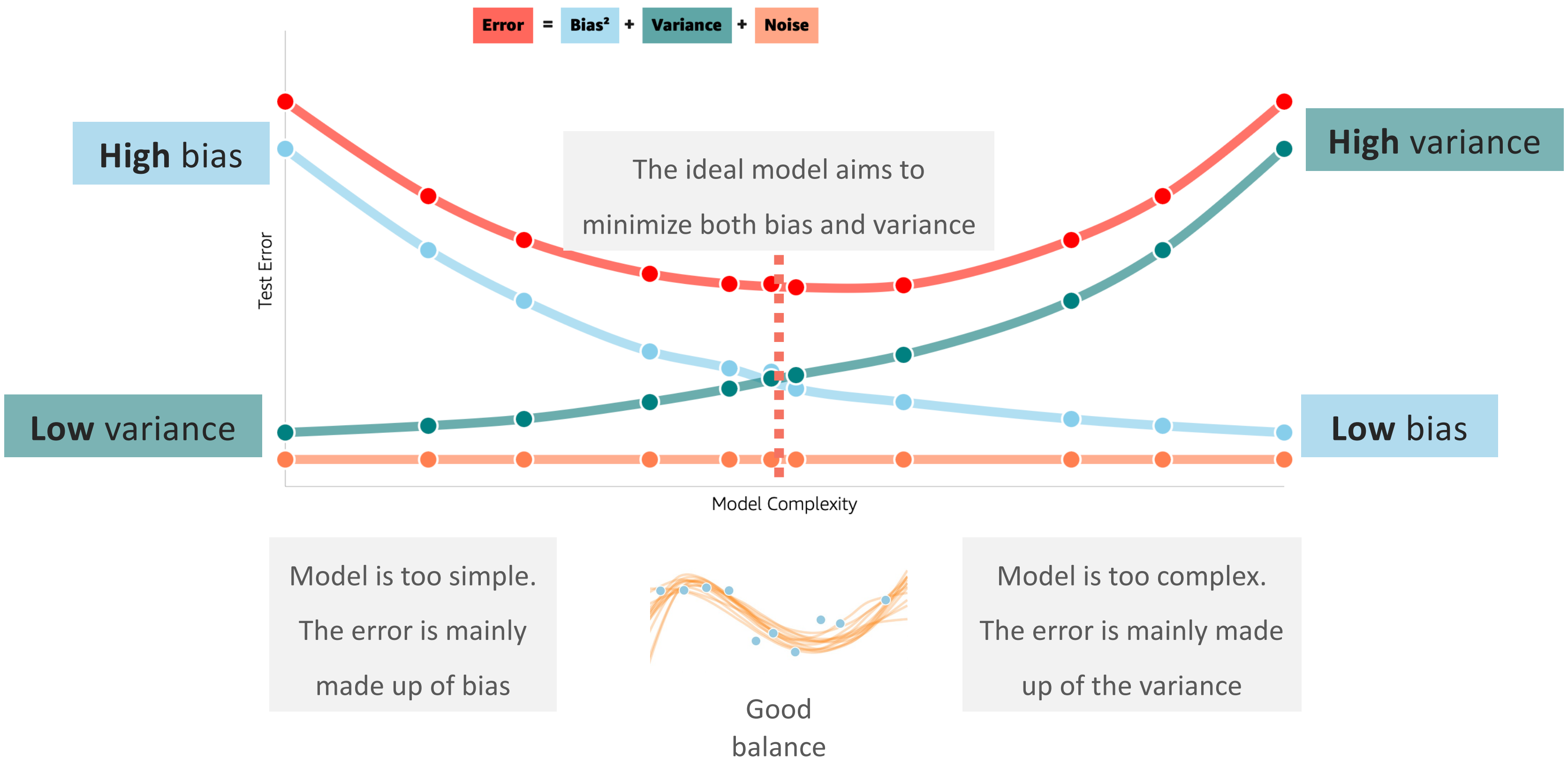
- Find a balance between **underfitting** (model too simple to learn meaningful patterns) and **overfitting** (model too complex to generalize to unseen data).
- By allowing a bit more variance (e.g., increasing model complexity) to reduce bias, we can aim for a model with optimal generalization performance.



Bias-Variance Tradeoff

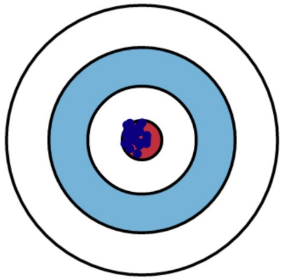
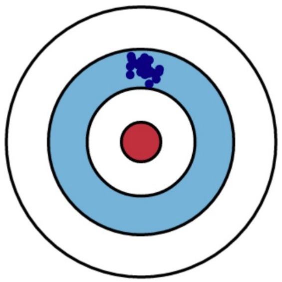
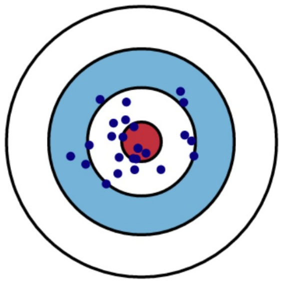
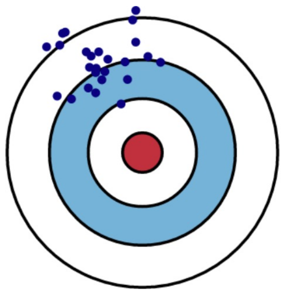
Erick Cuenca - twitter: @erickedu85

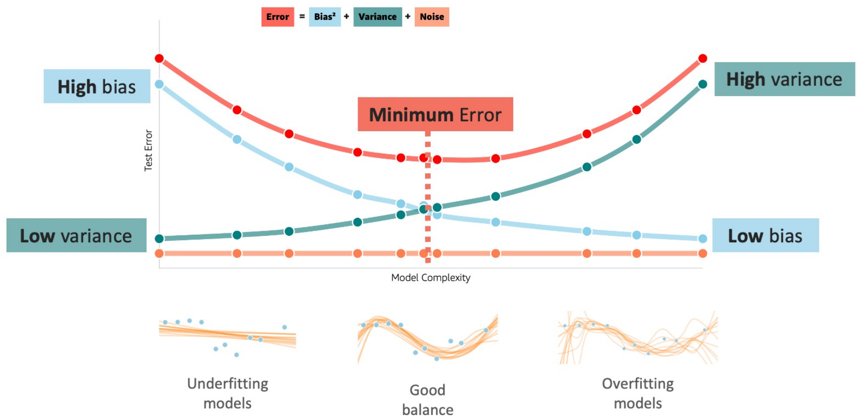
Bias-Variance Tradeoff



Erick Cuenca - twitter: @erickedu85

Bias-Variance Tradeoff

	Low Bias	High Bias
Low Variance	***Ideal***: accurate and consistent predictions (perfect fit) 	***Underfitting:*** predictions are far from the true value but clustered together 
High Variance	***Overfitting:*** some predictions are close, but highly scattered (model too sensitive) 	***Worst case***: predictions are both inaccurate and inconsistent 



Bias-Variance Tradeoff

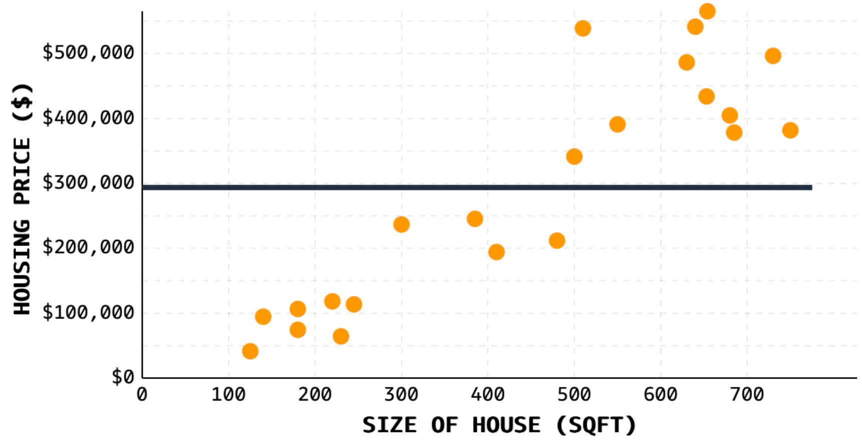
- Increasing bias reduce variance and vice-versa
- $Error = Bias^2 + Variance + Noise$
- The best model is where the error is reduced
- Compromise between bias and variance

Linear regression

Linear regression

Linear regression is a simple yet powerful method for predicting a numeric response from one or more independent variables.

- It is a **supervised algorithm** that learns to model a dependent variable, y, as a function of some independent variables (aka "**features**"), x_i, by finding a line (in 2D) or a hyperplane (in higher dimensions) that best "fits" the data.
- In general, we assume y to be some number and each x_i can be basically anything. For example:
  - Predicting the price of a house using the number of rooms in that house (y : price, x_1: number of rooms)
  - Predicting a person's weight based on their height and age (y : weight, x_1: height, x_2: age)



Linear regression

In general, the equation for linear regression is:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon$$

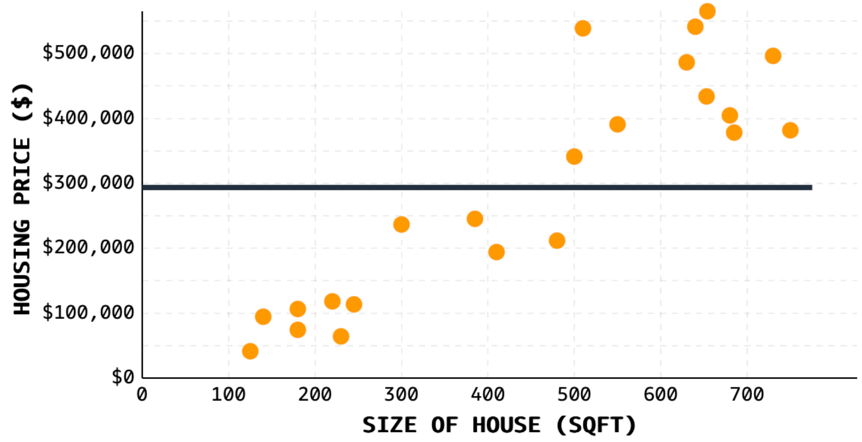
Where,

$\boldsymbol{y}$ = the dependent variable: the thing we are trying to predict

$\boldsymbol{x_i}$ = the independent variable: the features our model uses to model y.

$\boldsymbol{\beta_i}$ = the coefficients (aka “weights”) of our regression model. These are the foundations of our model. They are what our model “learns” during optimization

$\boldsymbol{\varepsilon}$= the irreducible error in our model. A term that collects together all the unmodeled parts of our data



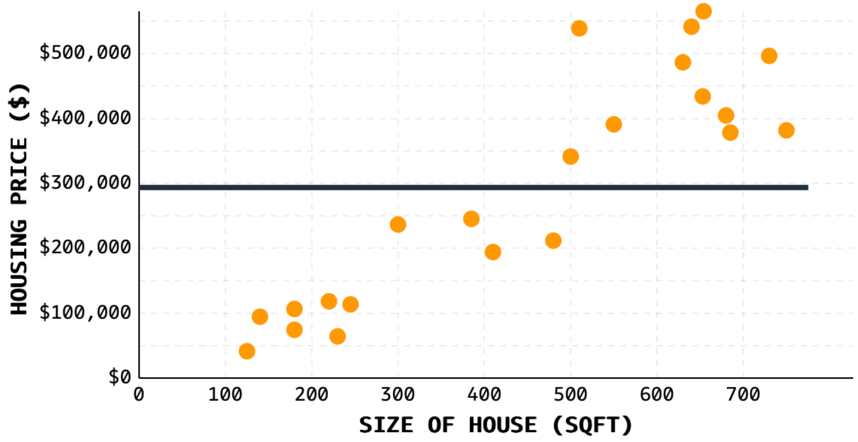
Linear regression

Fitting a linear regression model involves finding the set of coefficients that best approximate $\boldsymbol{y}$ as a function of the input features.

- Although the true parameters of the data-generating process are unknown, we can estimate them from our data.
- Once we've estimated these coefficients ($\hat{\beta}_i$), we can predict futures values ($\hat{y}$) using the equation:

$$\hat{\boldsymbol{y}} = \hat{\boldsymbol{\beta}}_0 + \hat{\boldsymbol{\beta}}_1 \boldsymbol{x}_1 + \hat{\boldsymbol{\beta}}_2 \boldsymbol{x}_2 + \cdots + \hat{\boldsymbol{\beta}}_p \boldsymbol{x}_p + \boldsymbol{\varepsilon}$$

- Predicting future values (often referred as *inference*) is as simple as plugging the values of our features x_i into the model equation



Linear regression

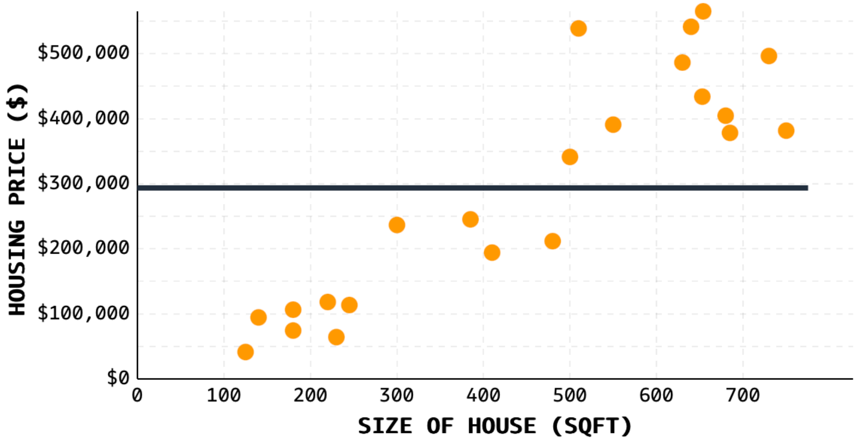
E.g., Let's fit a model to *predict* **housing price** ($) using the size of the house **sqft** (in square-footage):

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

$$house_{price} = \hat{\beta}_0 + \hat{\beta}_1 * sqft$$

We'll start with a very simple model, predicting the price of each house to be just the average house price in our dataset, ~$290,000, ignoring the different sizes of each house:

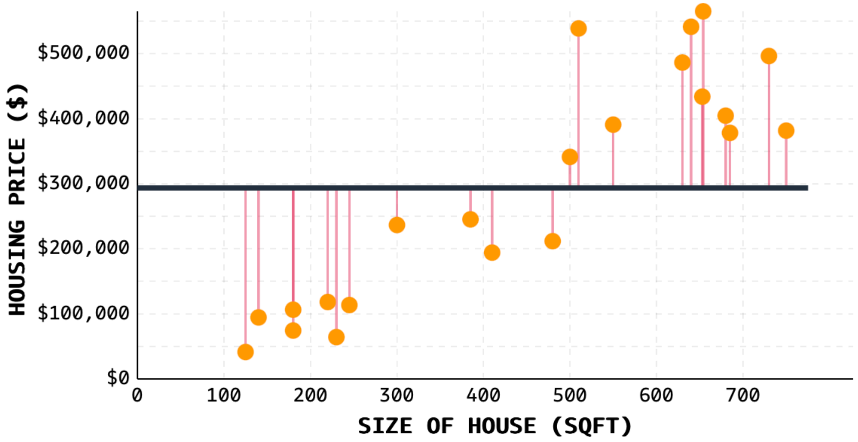
$$house_{price} = 290000 + 0 * 290000$$



Linear regression

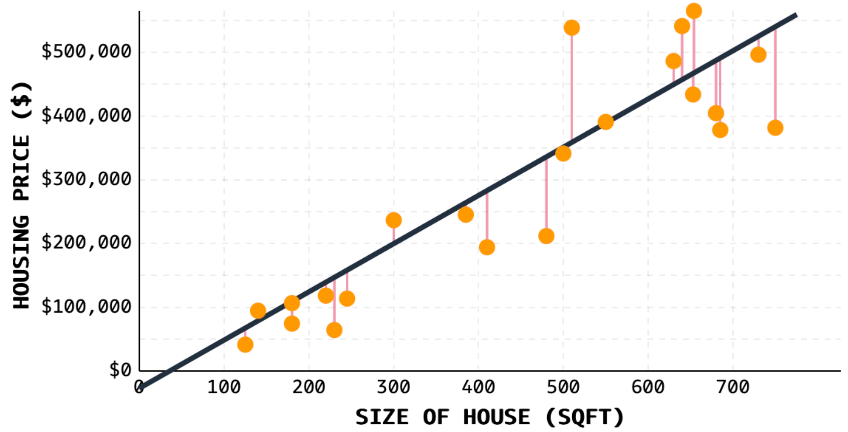
To **evaluate** our model's performance **quantitatively**, we plot the error for each observation. These errors, or **residuals**, represent the *distance between the actual observation and the predicted value* for that observation.

As shown in the plot, we can clearly see that our model exhibits a significant amount of error.



Linear regression

- The goal of linear regression is to reduce prediction error by finding a line or surface that best fits our data.
- For a simple regression problem, this involves estimating the ***intercept*** ($\hat{\beta}_0$) and ***slope*** ($\hat{\beta}_1$) that define the relationship between our input features and the target variable.



There is still some error, but the general pattern is well captured. As a result, we can be reasonably confident that, by plugging in new values for square footage, our predicted house prices will be reasonably accurate.

Linear regression

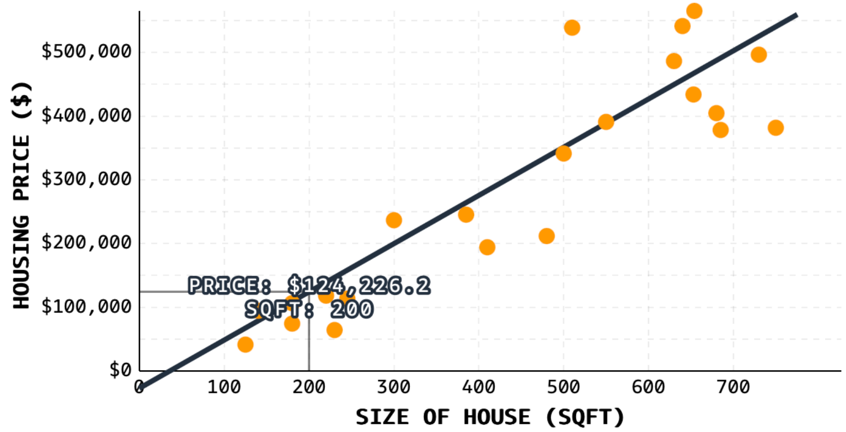
Predicting future values:

- We just plug in any x_i values into our equation!
- For our simple model, that means plugging in a value for **sqft** into our model:

$$\textbf{sqft}\ \text{value} = 200;$$

$$\hat{y} = -27153.8 + 756.9 * \mathbf{200}$$

$$\hat{y} = 124226$$



Our model predicts a house that is **200 square-feet** will **cost $124,226**

Linear regression

Predicting future values:

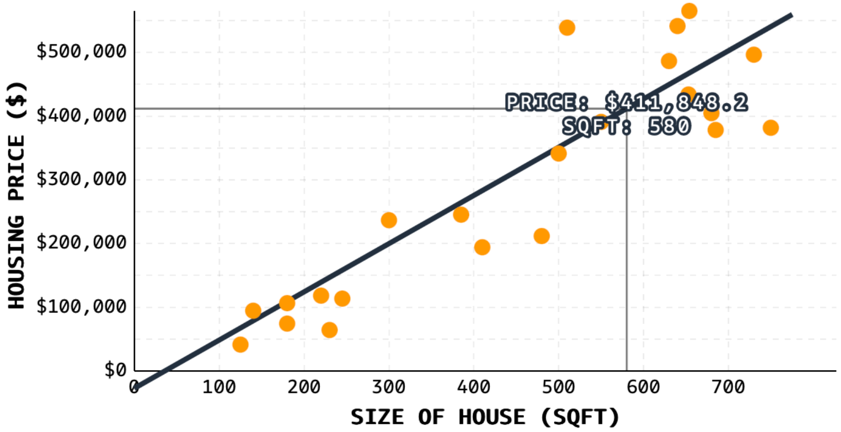
- We just plug in any $\boldsymbol{x_i}$ values into our equation!
- For our simple model, that means plugging in a value for **sqft** into our model:

$\mathbf{sqft}$ value $= 580;$

$\hat{y} = -27153.8 + 756.9 * \mathbf{580}$

$\hat{y} = 411848$

Our model predicts a house that is **580 square-feet** will **cost $411,848**



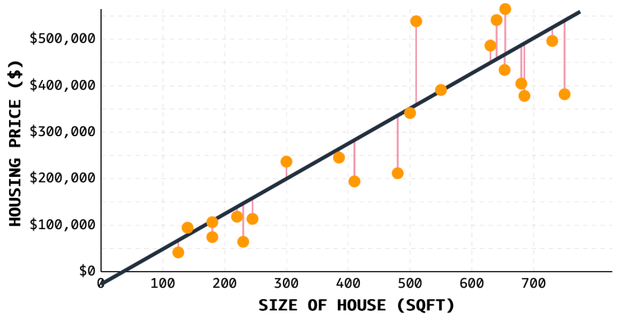
Model evaluation

They quantified how close the predicted value is to the true value. We'll fit our regression model to a set **training** data, and evaluate it's performance on the **test** dataset.

Error	Formula	Range	Description
Total Absolute Error **TAE**	$\text{TAE} = \sum_{i=1}^{n} \lvert Y_i - \hat{Y}_i \rvert$	≥ 0 (0 good)	• It is the sum of the absolute values of the difference between each true value and the predicted value (given by the regression line)
Mean Absolute Error **MAE**	$MAE = \frac{1}{n}\sum_{i=1}^{n} \lvert Y_i - \hat{Y}_i \rvert$	≥ 0 (0 good)	• It is the average of all absolute errors • Easy to interpreter: the units of MAE are the original of the data • **Not sensitive to outliers**: Treat larger and small errors equally
Mean-Squared Error **MSE**	$MSE = \frac{1}{n}\sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$	≥ 0 (0 good)	• It square the distance between each true value and the predicted value (given by the regression line), summing the squared values, and then dividing by the number of data points • **Sensitive to outliers**: It punishes large errors more

$n = number\ of\ data\ records$
$Y_i = true\ values$
$\hat{Y}_i = predicted\ values$

$Y = mean\ value\ of\ true\ values; Y = \frac{1}{n}\sum_{i=1}^{n} Y_i$



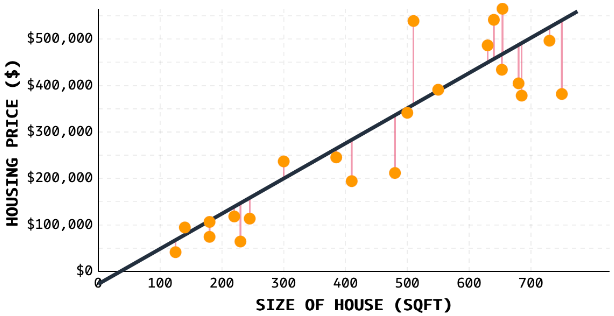
Model evaluation

They quantified how close the predicted value is to the true value. We'll fit our regression model to a set **training** data, and evaluate it's performance on the **test** dataset.

Error	Formula	Range	Description
Root Mean-Squared Error **RMSE**	$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}$	≥ 0 (0 good)	• It is the root of the Mean-Squared Error (MSE) • Easy to interpreter: the units of RMSE are the original of the data • Sensitive to outliers: It punishes large errors more
Root Squared Error $\mathbf{R^2}$	$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$	$0 \leq R^2 \leq 1$ (0 bad, 1 good)	• It summarizes how well a model fits the data. • It represents the % of the variance in $\mathbf{y}$ explained by the features $\mathbf{x}$. It takes the MSE (numerator) and the variance of the mean (denominator) • $R^2 = 1$: model capture 100% of the variance • R^2 Negative value: the predicted model performs worse than a simple average of the original data

$n = number\ of\ data\ records$
$Y_i = true\ values$
$\hat{Y}_i = predicted\ values$

$Y = mean\ value\ of\ true\ values; Y = \frac{1}{n}\sum_{i=1}^{n} Y_i$



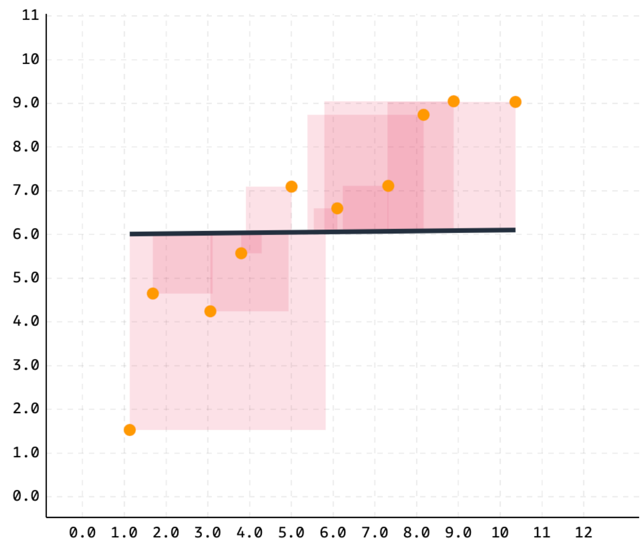
Model evaluation

$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$
$\boldsymbol{\hat{\beta}_0 = 6}$
$\boldsymbol{\hat{\beta}_1 = 0.01}$

$\hat{y} = 6 + 0.01x$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 = \mathbf{5.22}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2} = \mathbf{0.01 = 1\%}$$



Model evaluation

$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$
$\boldsymbol{\hat{\beta}_0 = 2}$
$\boldsymbol{\hat{\beta}_1 = 0.76}$

$\hat{y} = 2 + 0.76x$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 = \mathbf{0.71}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2} = \mathbf{0.87 = 87\%}$$