Machine Learning Foundations - Part 3

- by Nagaraju Budigam

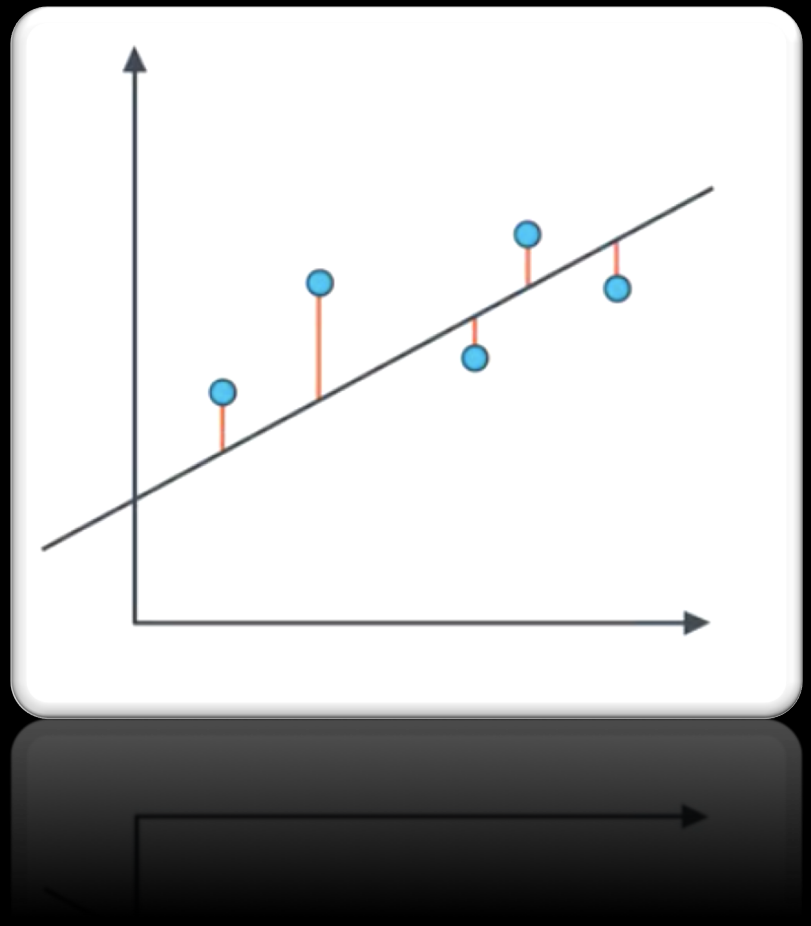
05/08/19

Learning Goals

- ❑ Regression Metrics
  - ✓ Mean Absolute Error
  - ✓ Mean Squared Error
- ❑ Error Detection
  - ✓ Underfitting
  - ✓ Overfitting
  - ✓ Model Complexity Graph
  - ✓ Learning Curves
  - ✓ K Fold Cross Validation

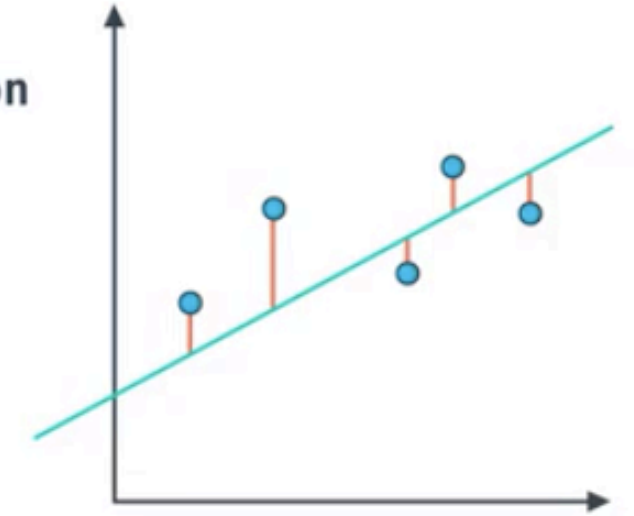
Regression Metrics – Mean Absolute Error

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - y_j|$$



Regression Metrics – Mean Absolute Error

```
from sklearn.metrics import mean_absolute_error
from sklearn.linear_model import LinearRegression

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

guesses = classifier.predict(X)

error = mean_absolute_error(y, guesses)
```

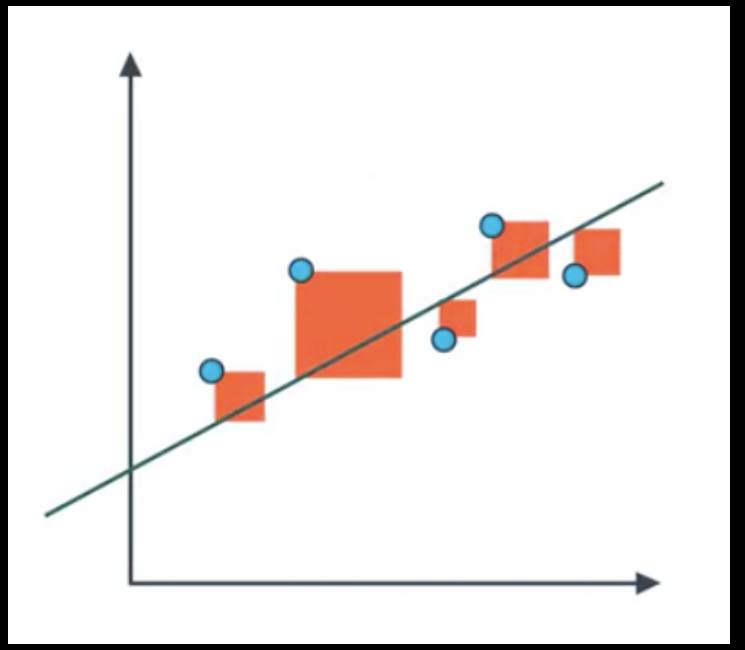


Regression Metrics – Mean Absolute Error

- Pros
  - ✓ Less sensitive to outliers
  - ✓ Many Small errors is same as a large error.
- Cons
  - Absolute Function is not differentiable.
  - Not a good option you want to use Gradient Descent as a error metric.

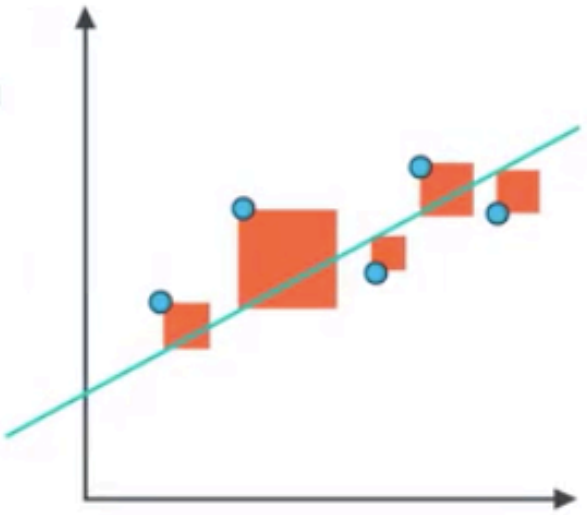
Regression Metrics – Mean Squared Error

$$\mathrm{MSE} = \sqrt{\frac{\sum_{i=1}^{n} \left(P_i - O_i\right)^2}{n}}$$



Regression Metrics – Mean Squared Error

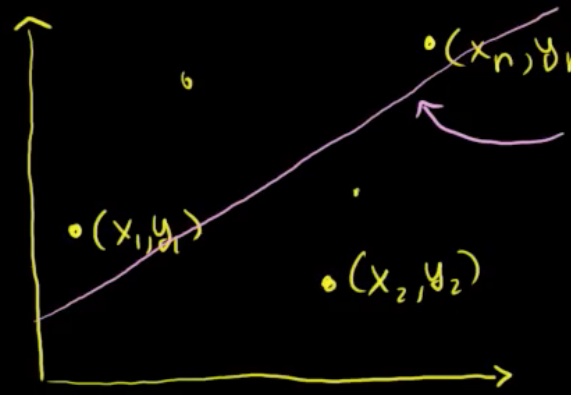
```
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression

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

guesses = classifier.predict(X)

error = mean_squared_error(y, guesses)
```



Regression Metrics – Mean Squared Error

- Pros
  - ✓ Emphasizes extremes than small errors
  - ✓ Square Function is differentiable.
  - ✓ Good option you want to use Gradient Descent as a error metric.
- Cons
  - Very sensitive to outliers.

Regression Metrics – R2 Score (Coefficient of Determination)



$$SE_{LINE} = (y_1 - (mx_1 + b))^2 + (y_2 - (mx_2 + b))^2 + \dots + (y_n - (mx_n + b))^2$$

total variation in y : $(y_1 - \bar{y})^2 + (y_2 - \bar{y})^2 + \dots + (y_n - \bar{y})^2 = SE_{\bar{y}}$

$\frac{SE_{line}}{SE_{\bar{y}}}$ How much of total variation is not described by the regression line?

$1 - \frac{SE_{line}}{SE_{\bar{y}}}$ = what % of total variation is desc by the variation in

Regression Metrics – R2 Score (Coefficient of Determination)

BAD MODEL

The errors should be similar.

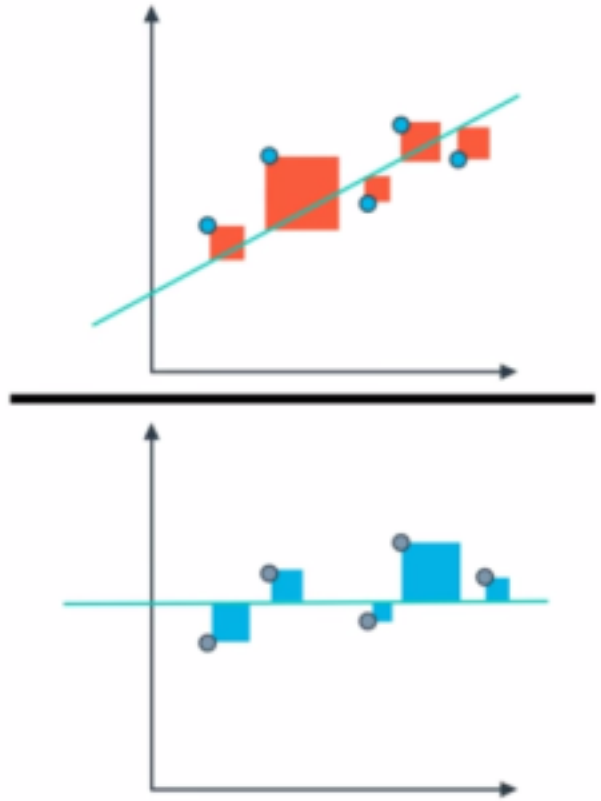
R2 score should be close to 0.

GOOD MODEL

The mean squared error for the linear regression model should be a lot smaller than the mean squared error for the simple model.

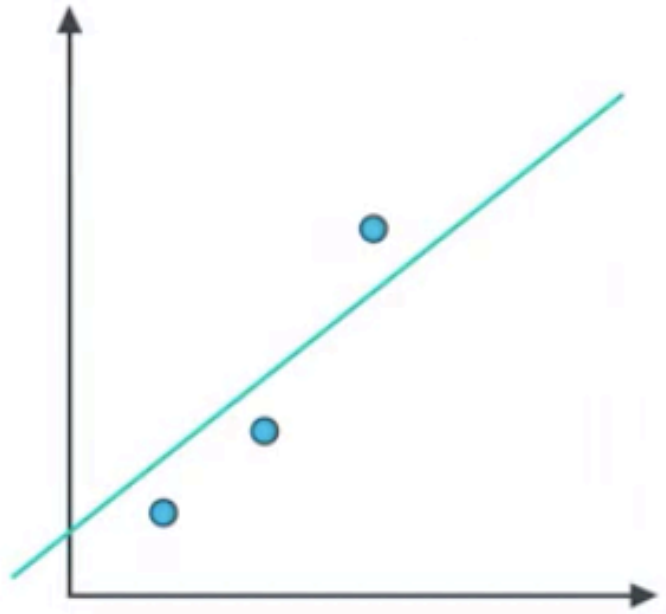
R2 score should be close to 1.

$$R2 = 1 - \frac{}{}$$



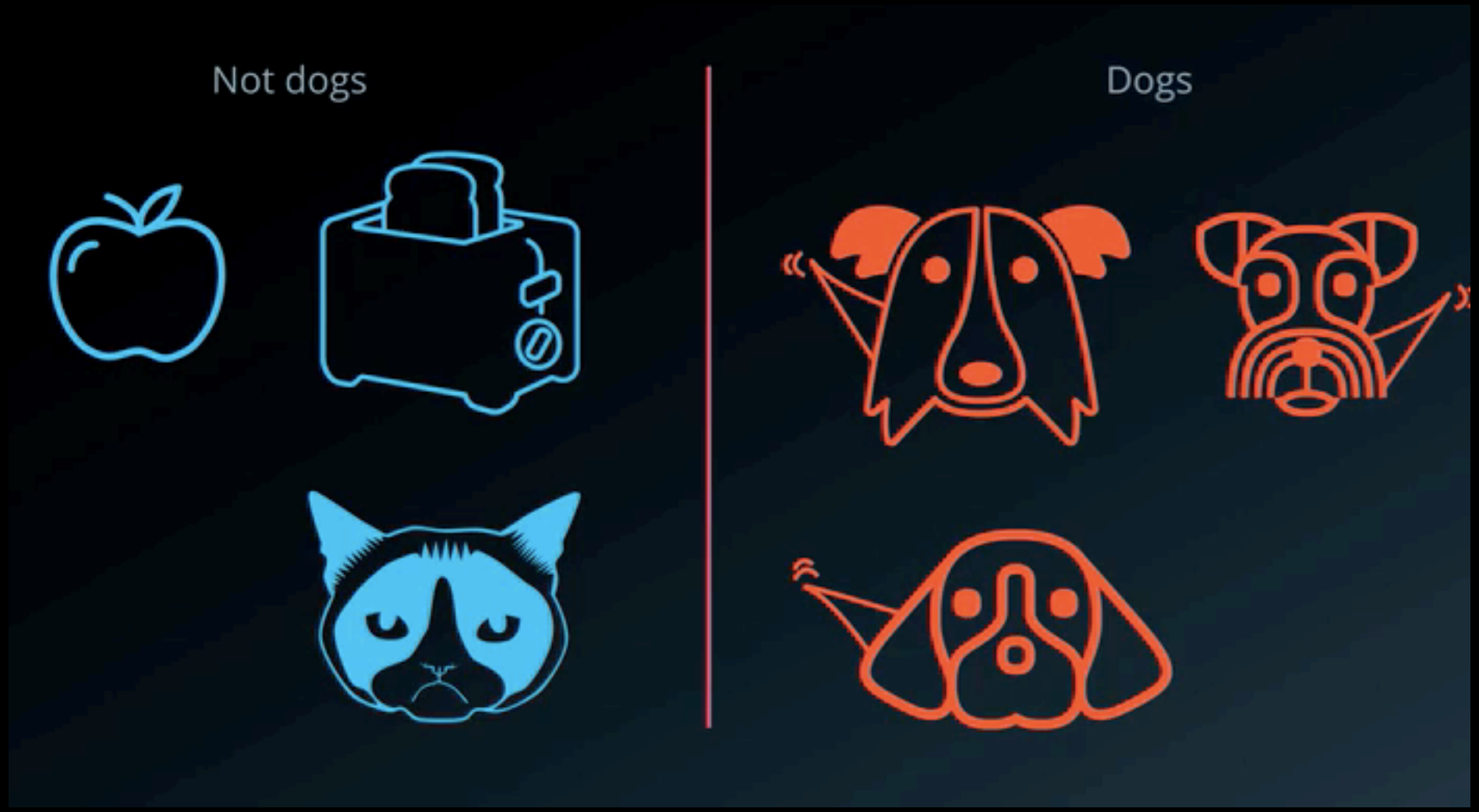
Regression Metrics – R2 Score (Coefficient of Determination)

```
from sklearn.metrics import r2_score

y_true = [1, 2, 4]
y_pred = [1.3, 2.5, 3.7]

r2_score(y_true, y_pred)
```

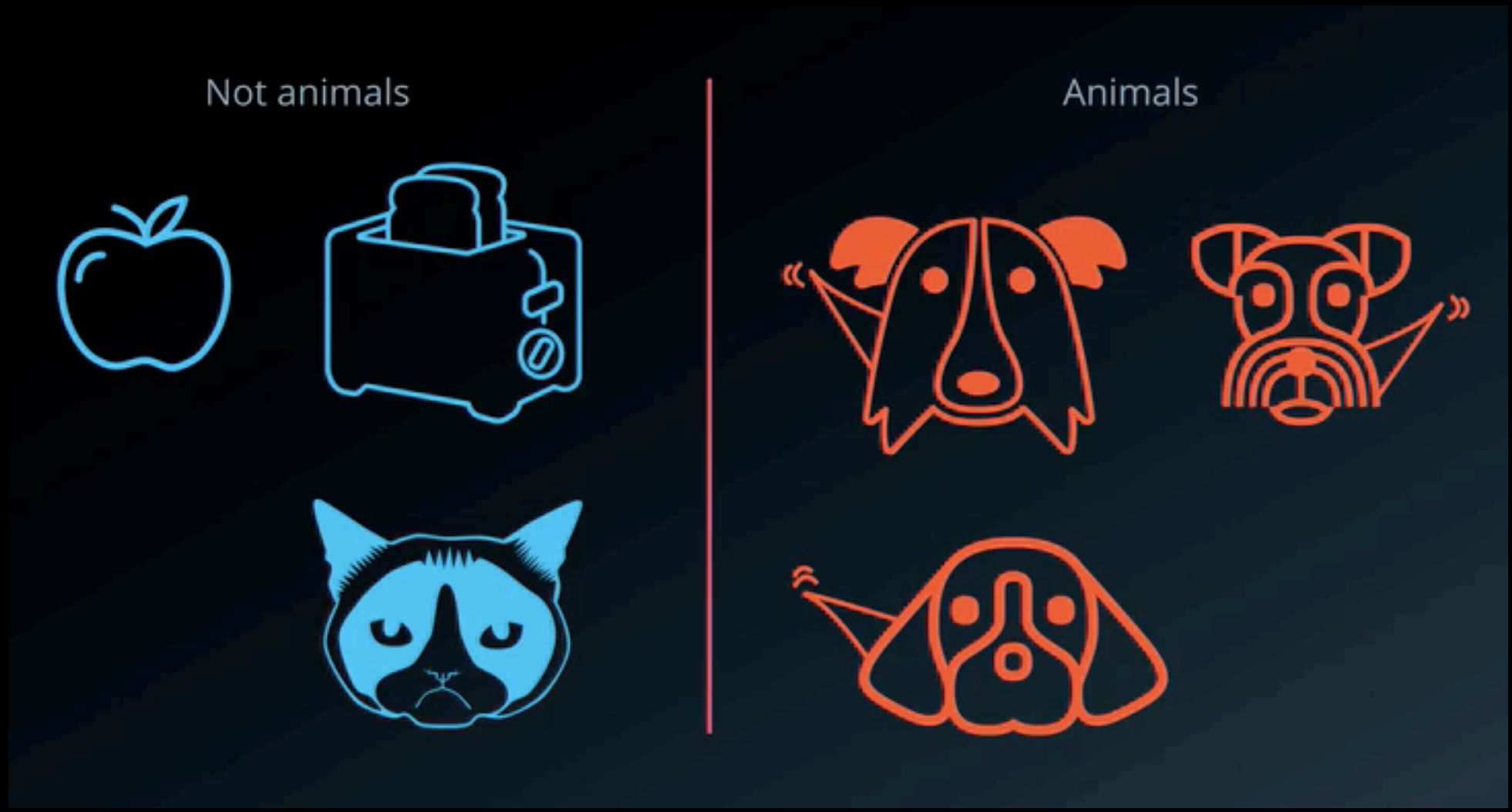


Error Detection – Types of Errors

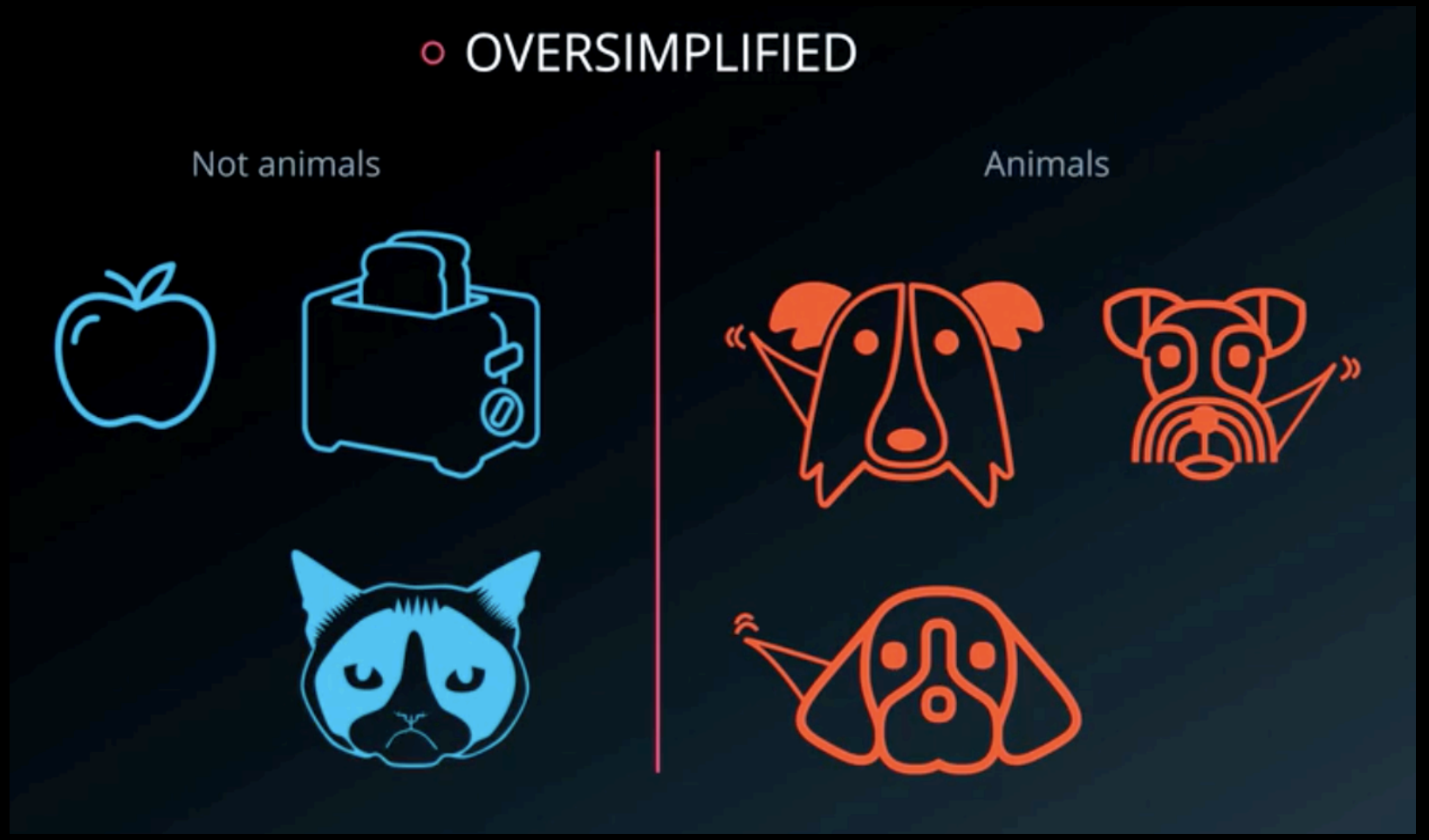
Error Detection – Types of Errors



Error Detection – Types of Errors



Error Detection – Types of Errors



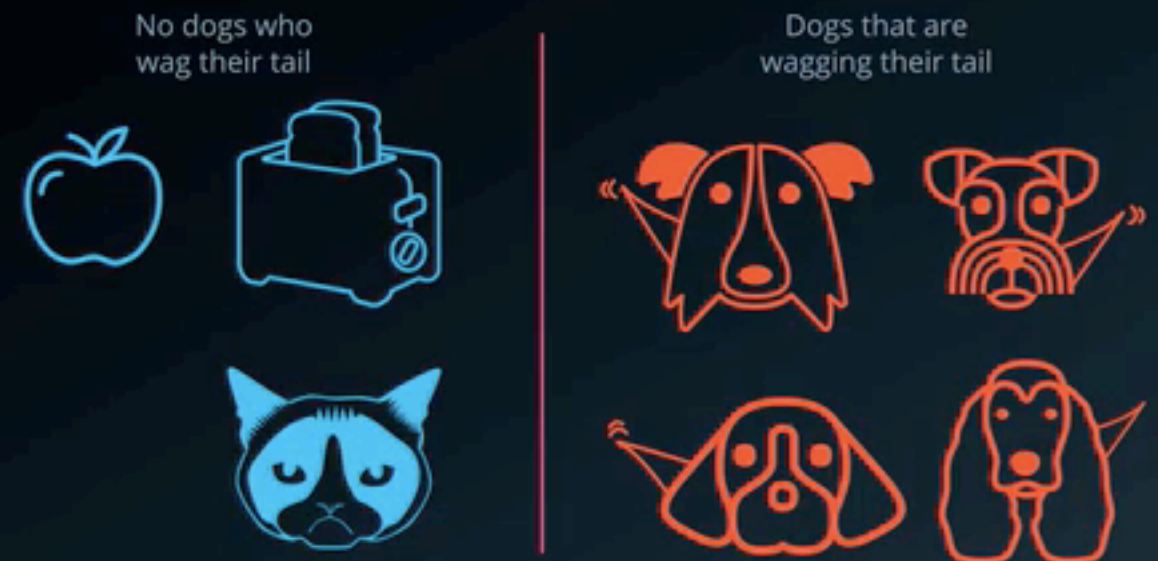
Error Detection – Types of Errors - Underfitting

- Does not do well in the training set.
- Error due to bias.

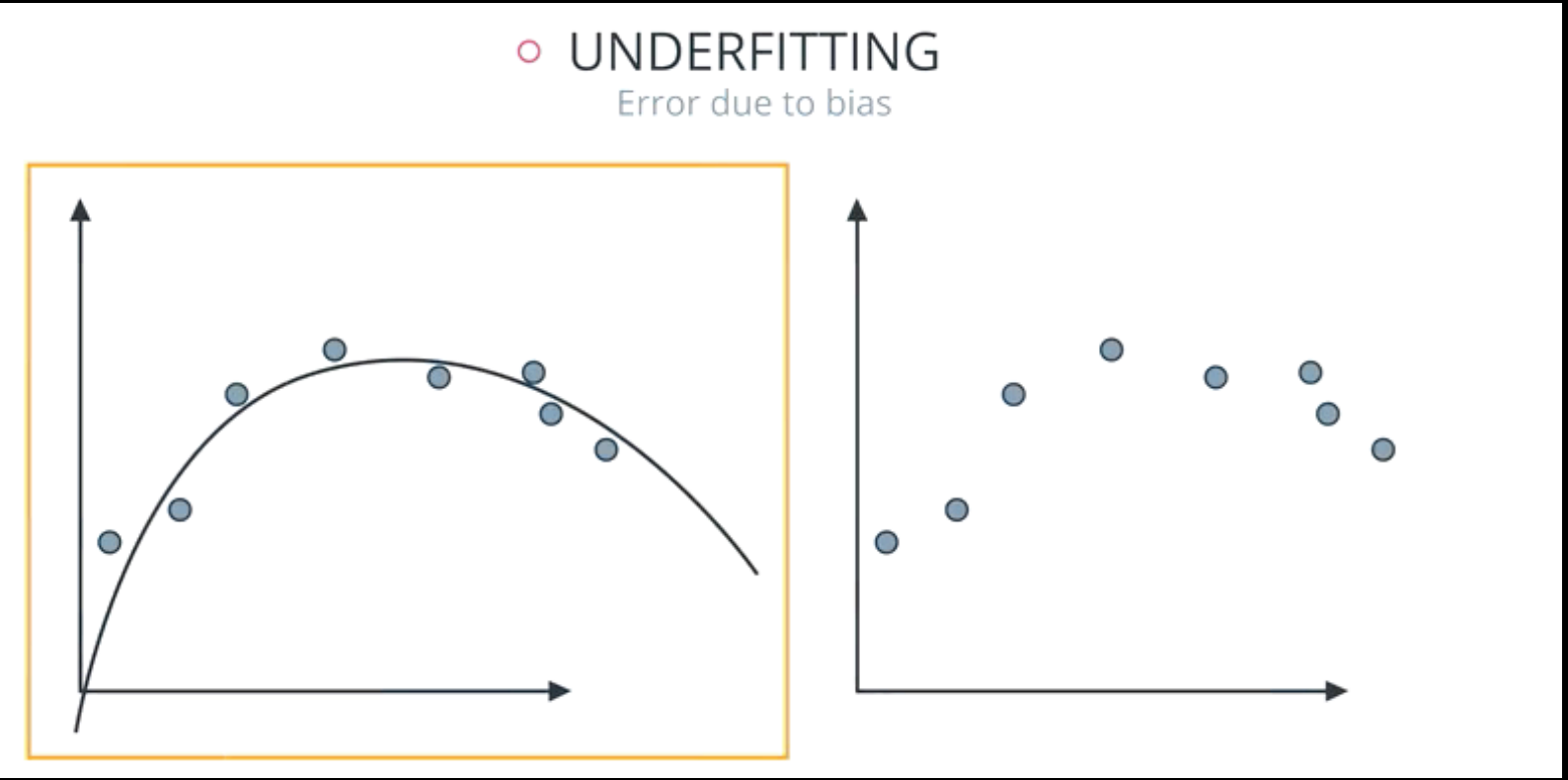
Error Detection – Types of Errors

Error Detection – Types of Errors - Overfitting

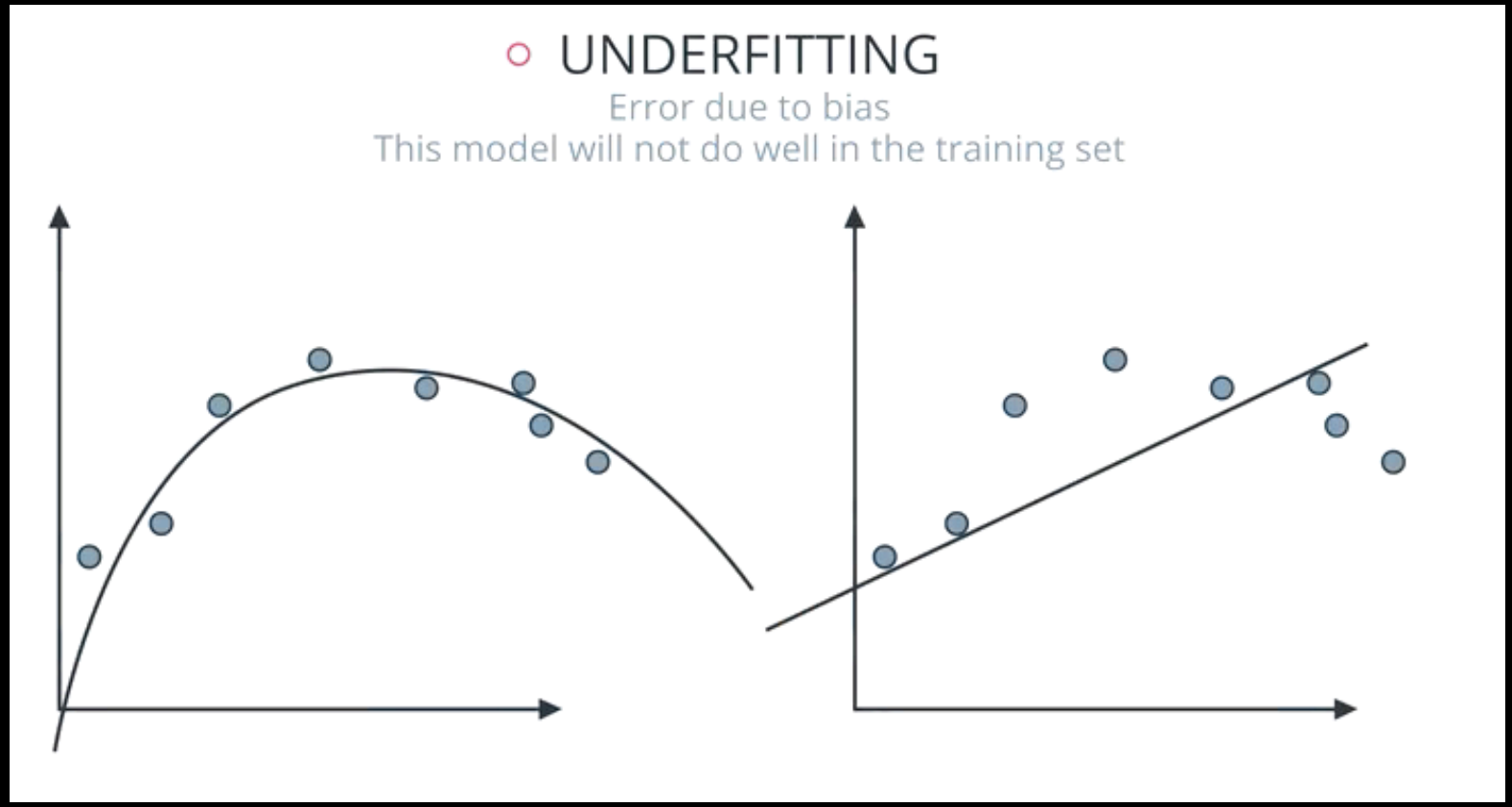
- Does well in the training set, but it tends to memorize it instead of learning the characteristics of it.
- Error due to variance.



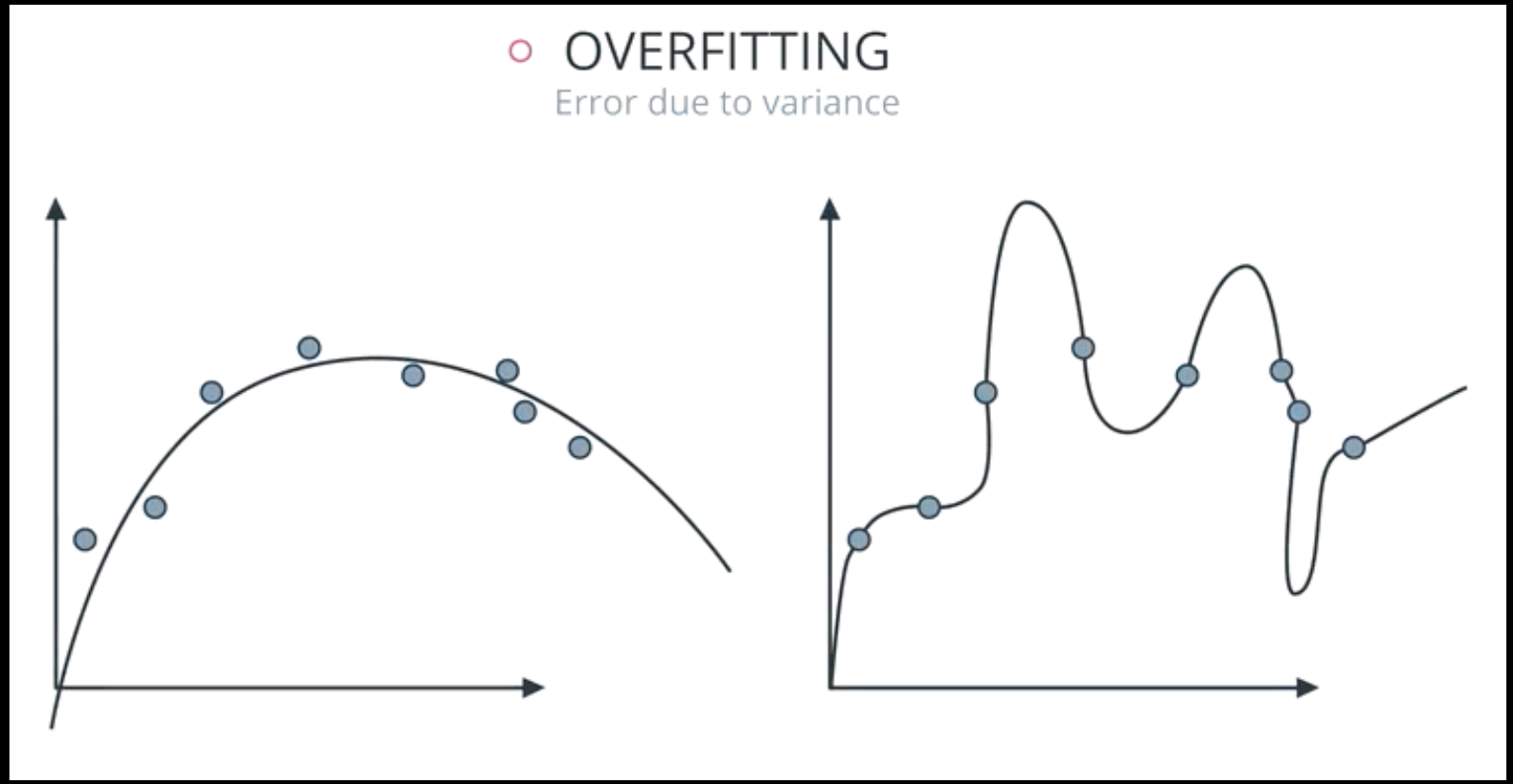
Error Detection – Underfitting



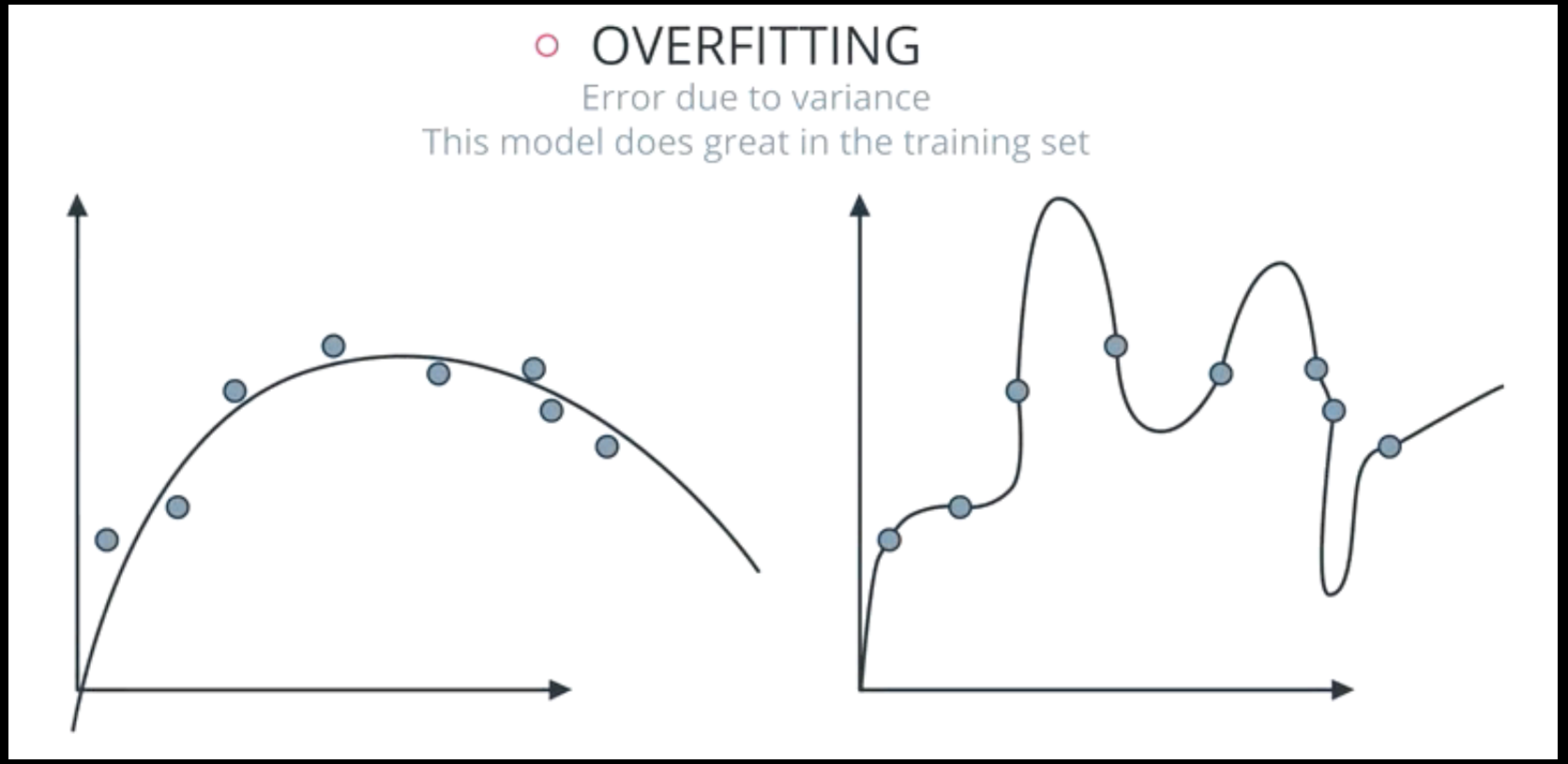
Error Detection – Underfitting



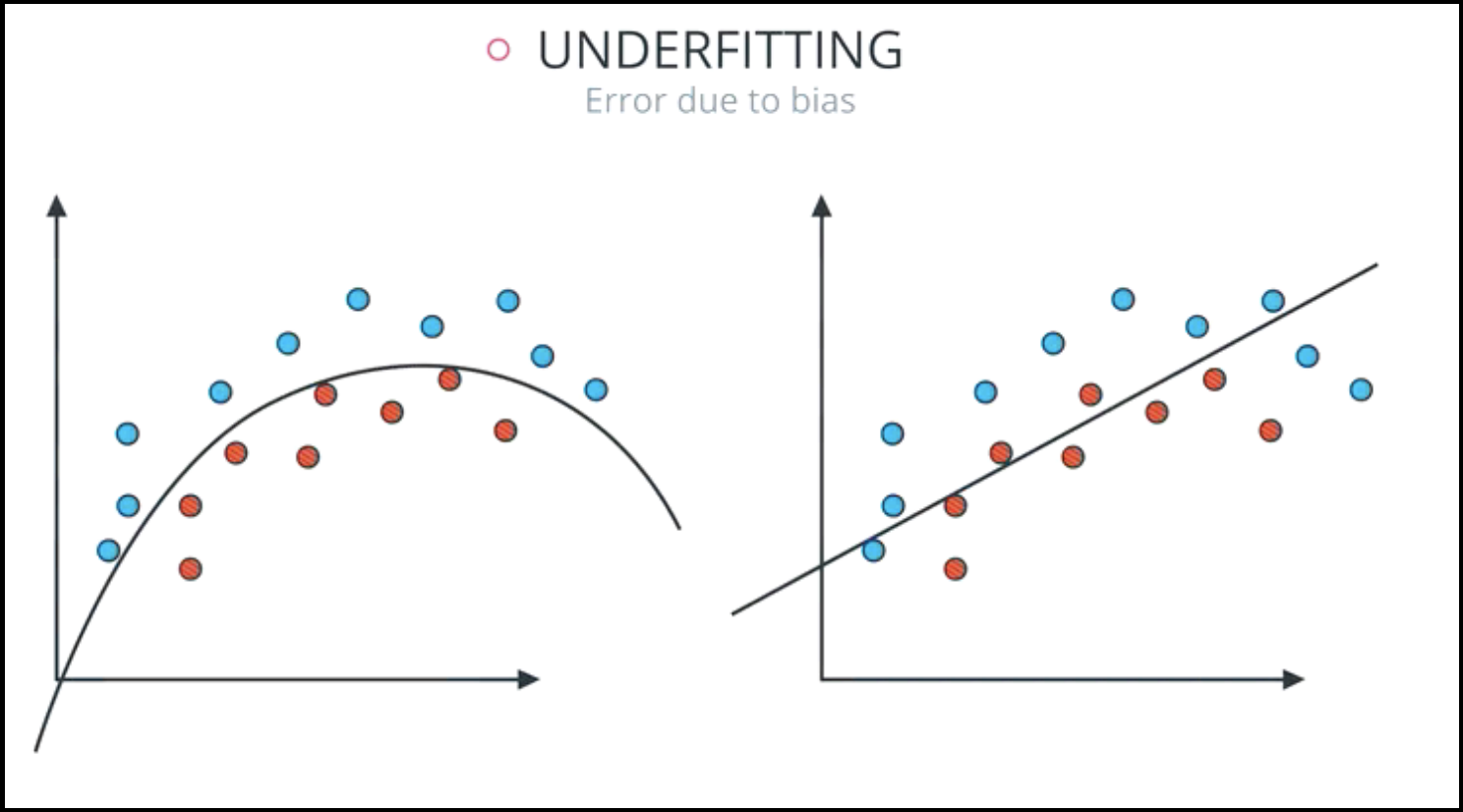
Error Detection – Overfitting



Error Detection – Overfitting

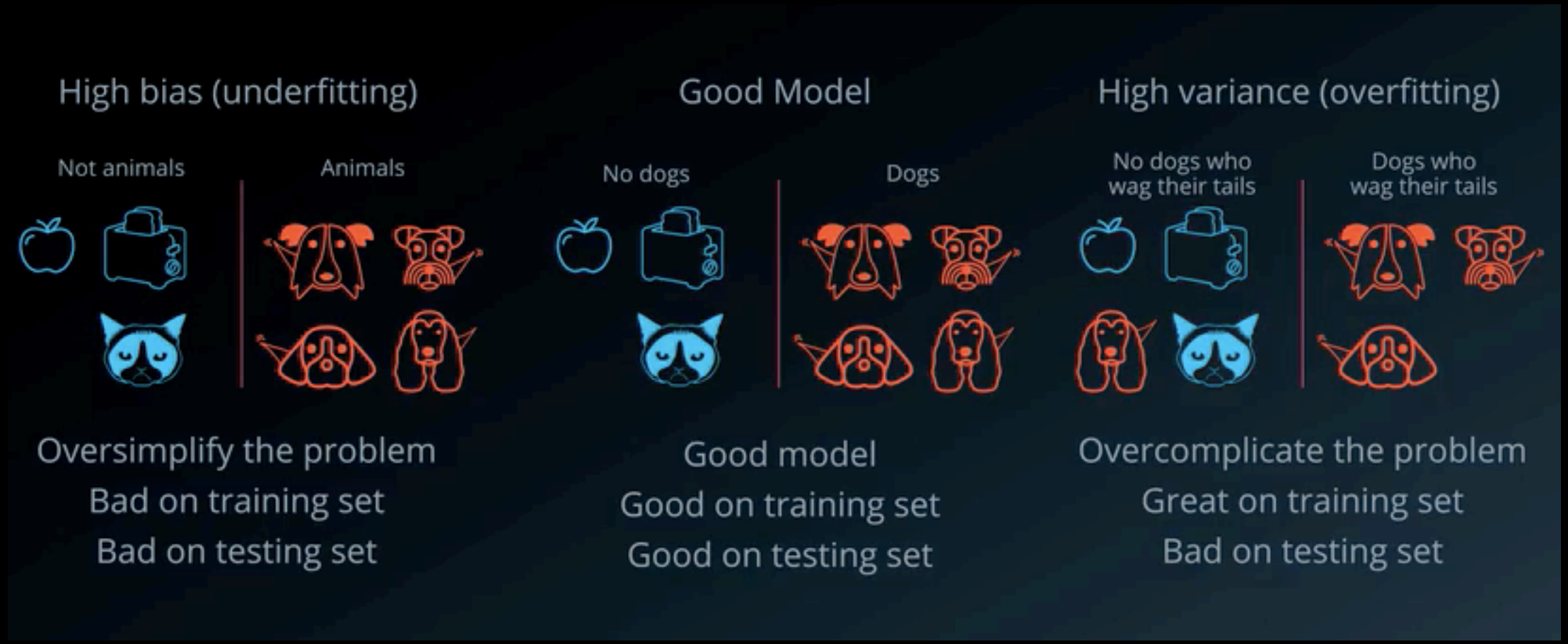


Error Detection – Underfitting

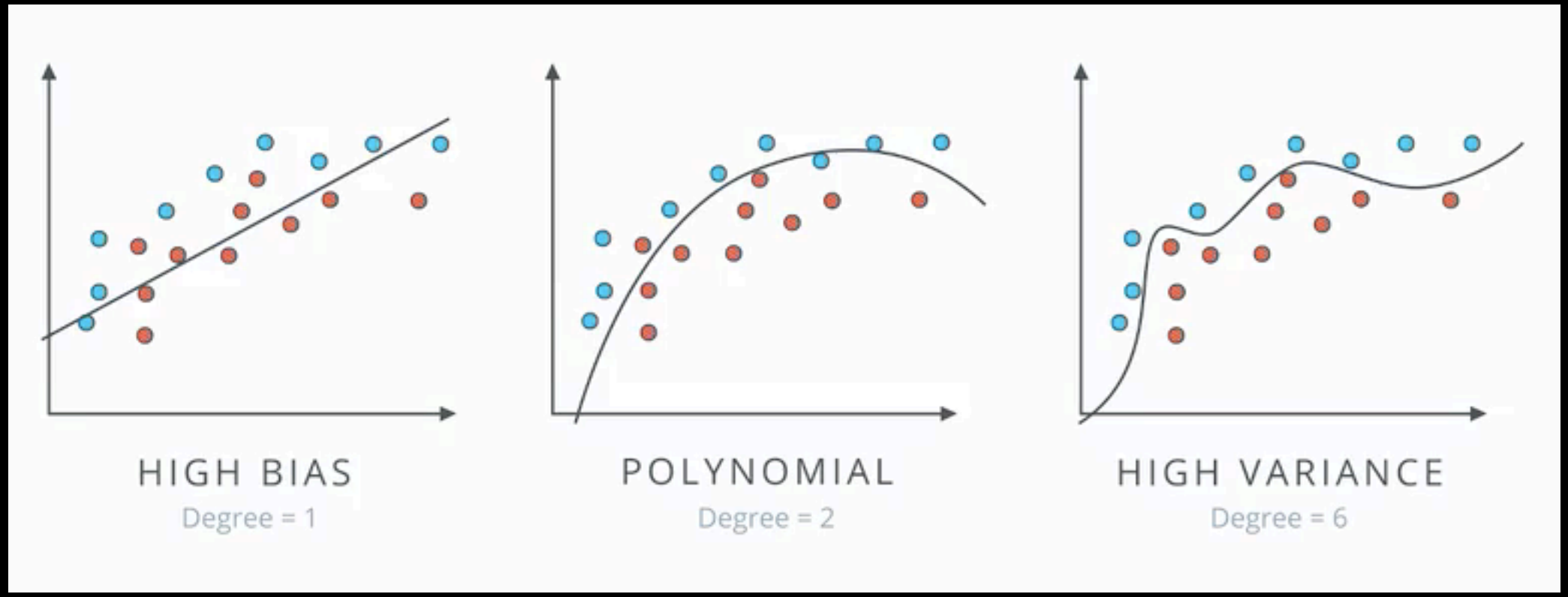


Error Detection – Overfitting

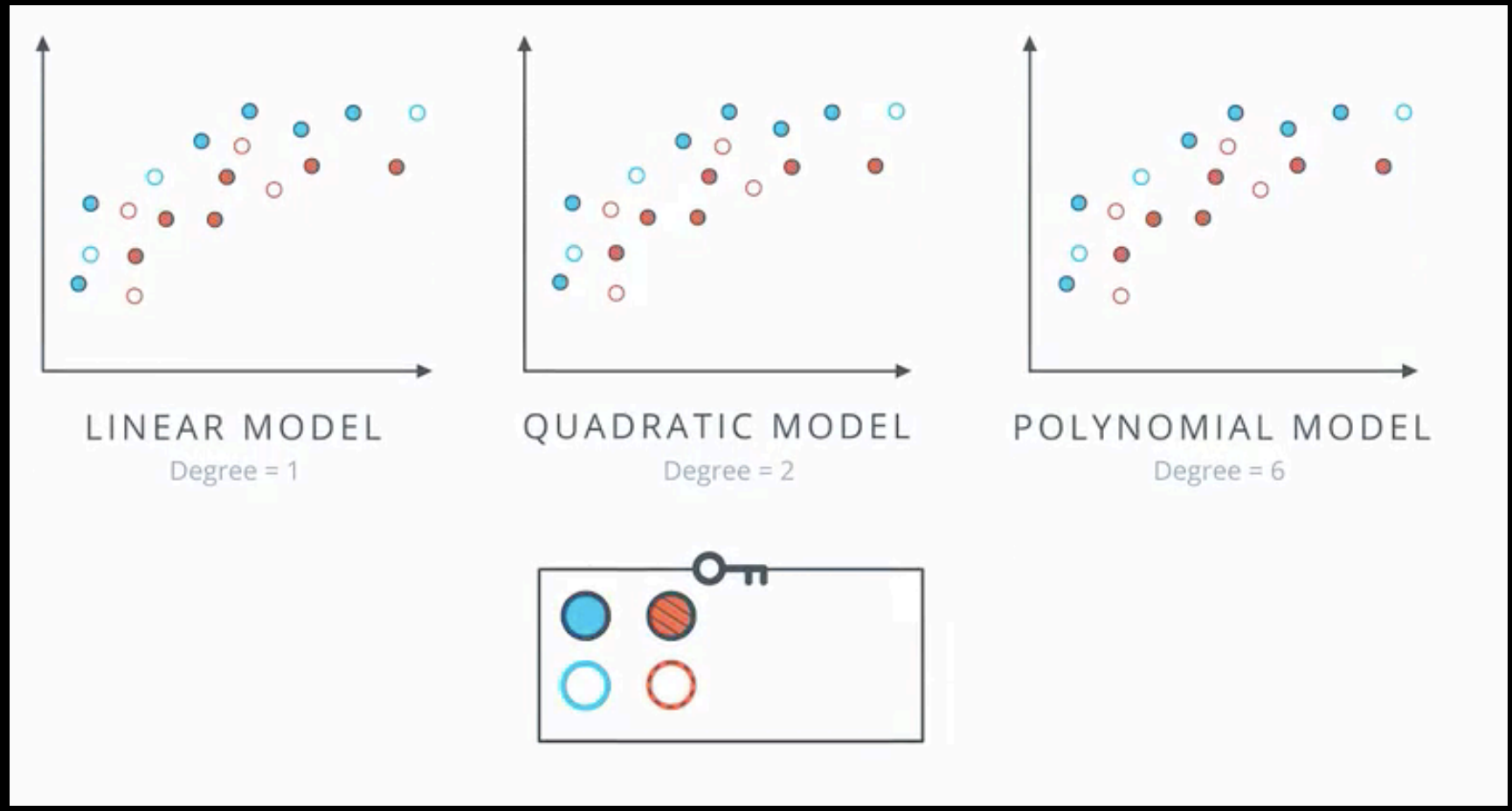
Error Detection – Summary



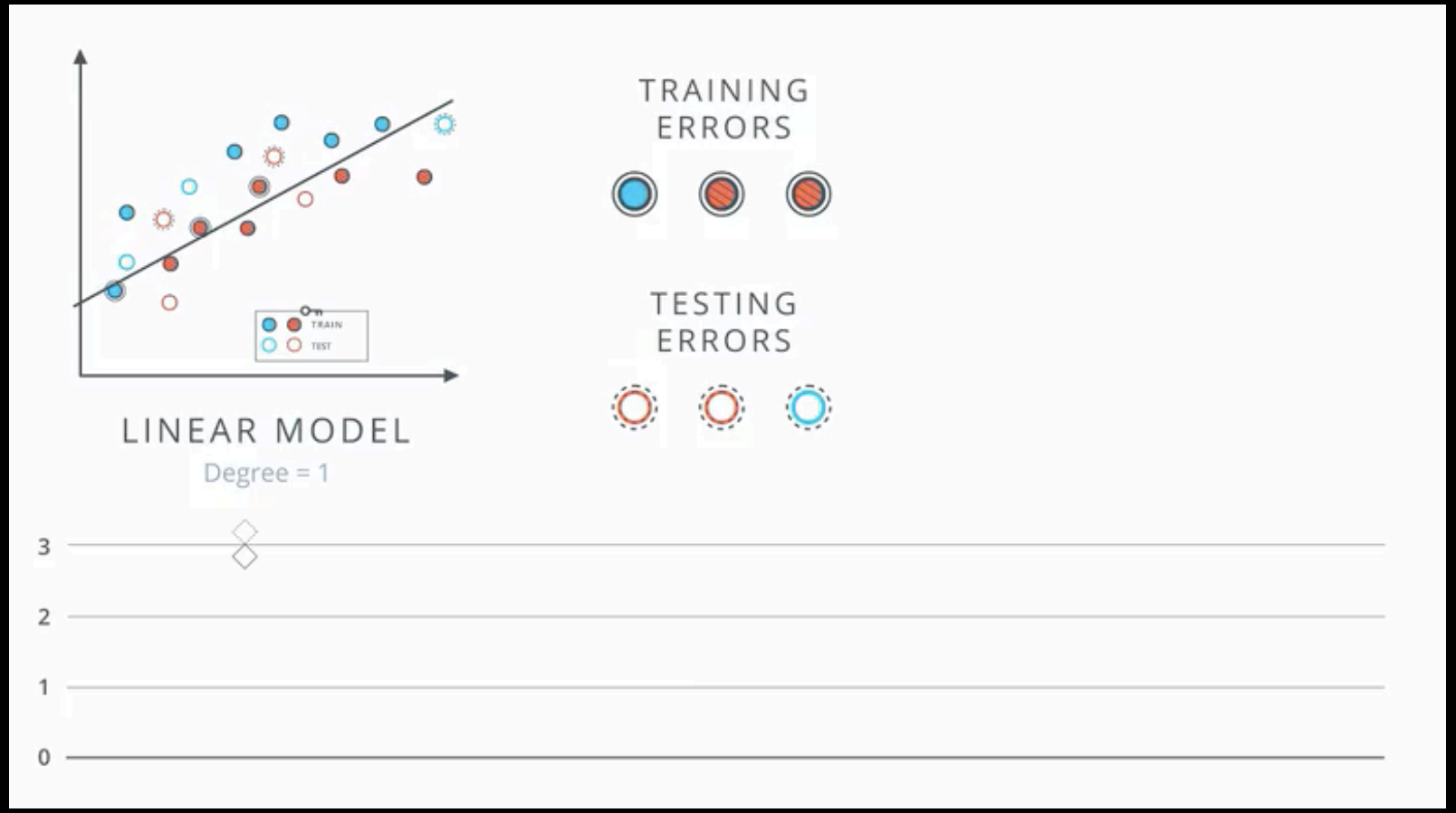
Model Complexity Graph



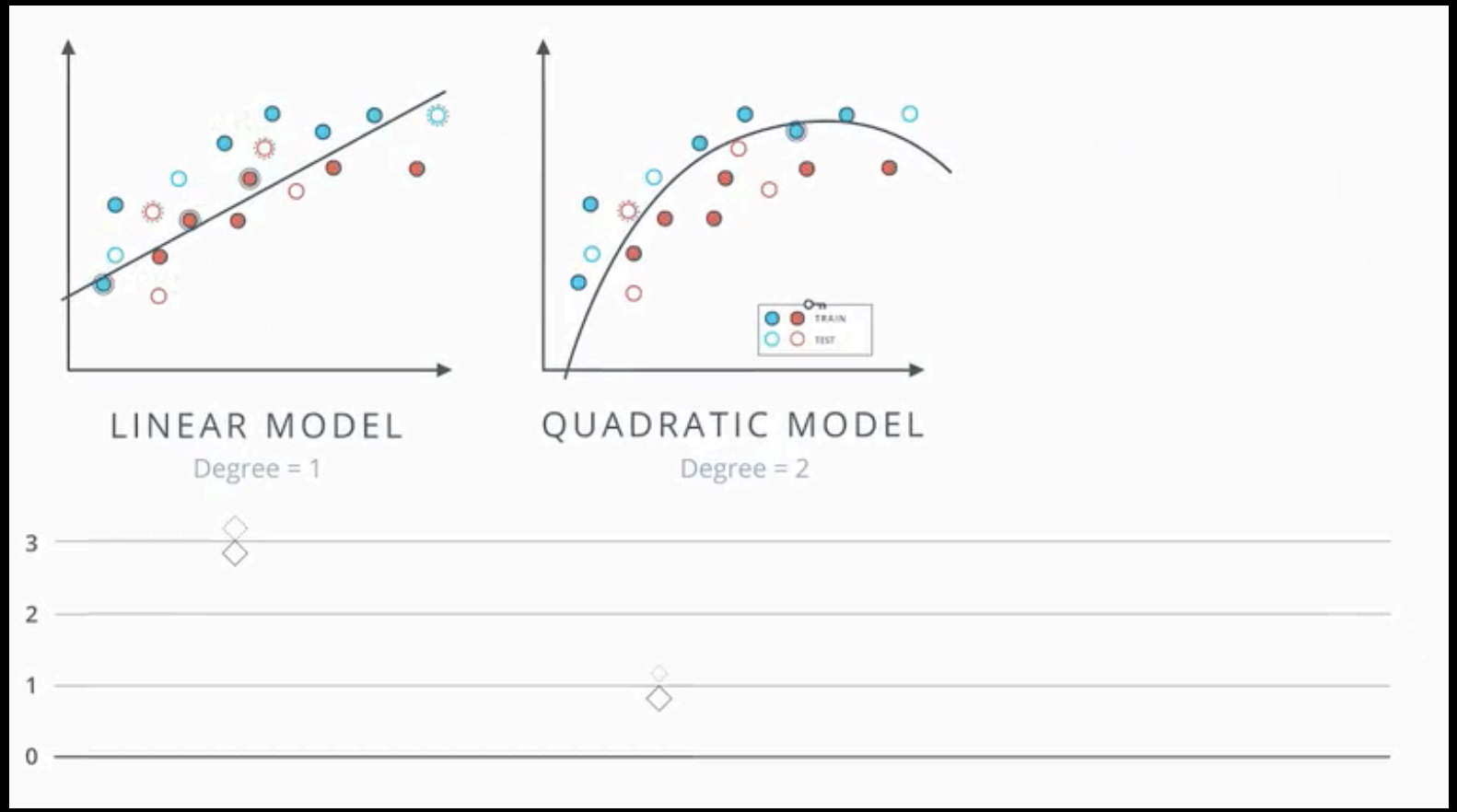
Model Complexity Graph



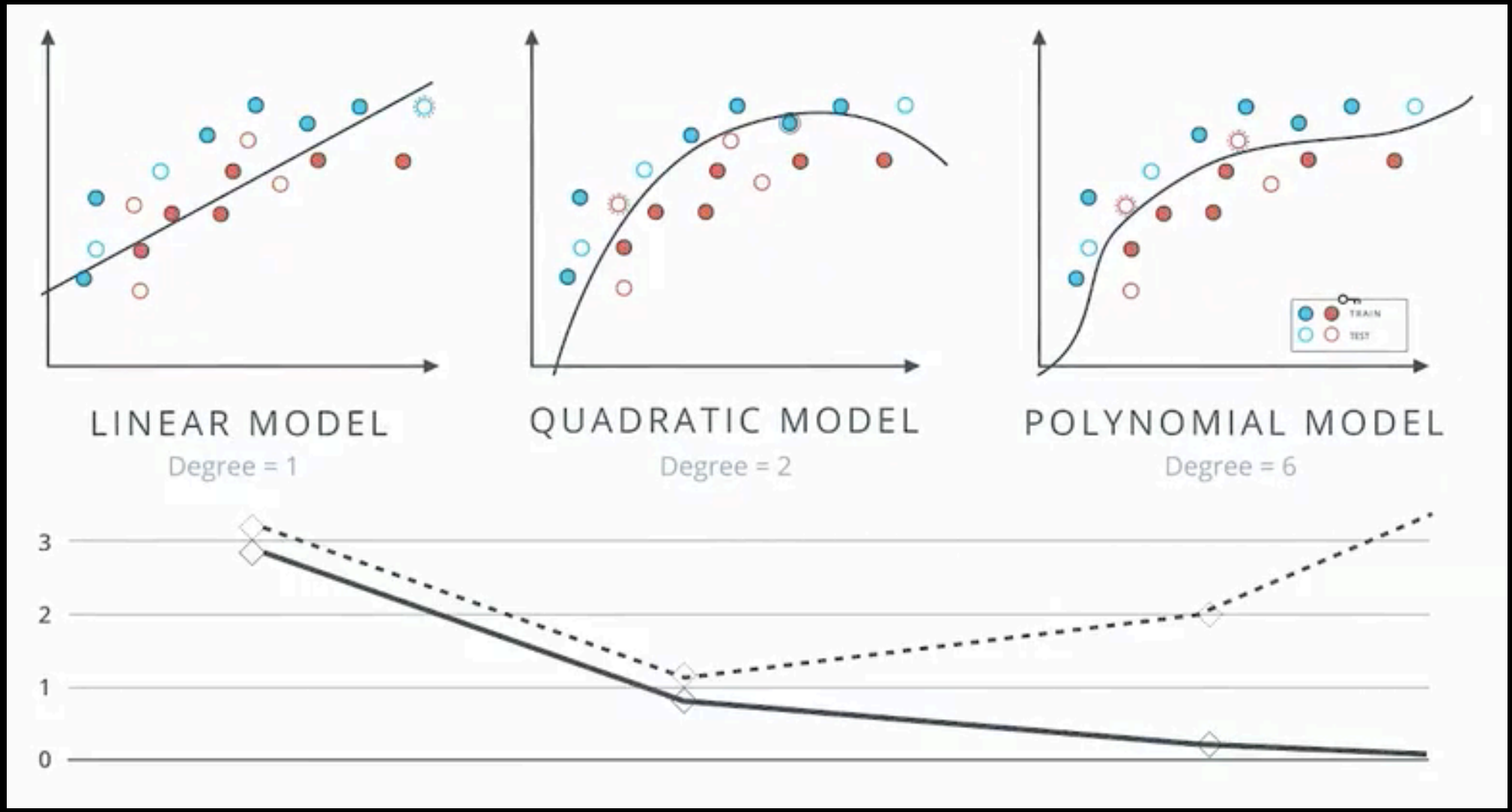
Model Complexity Graph



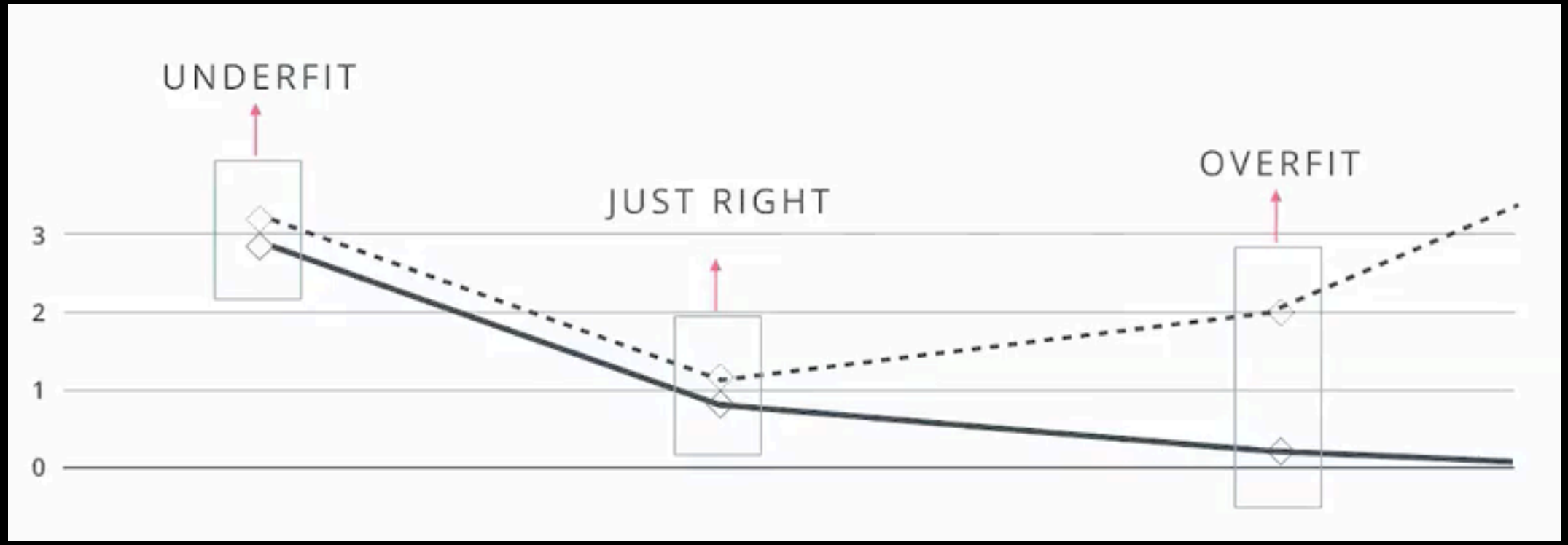
Model Complexity Graph



Model Complexity Graph



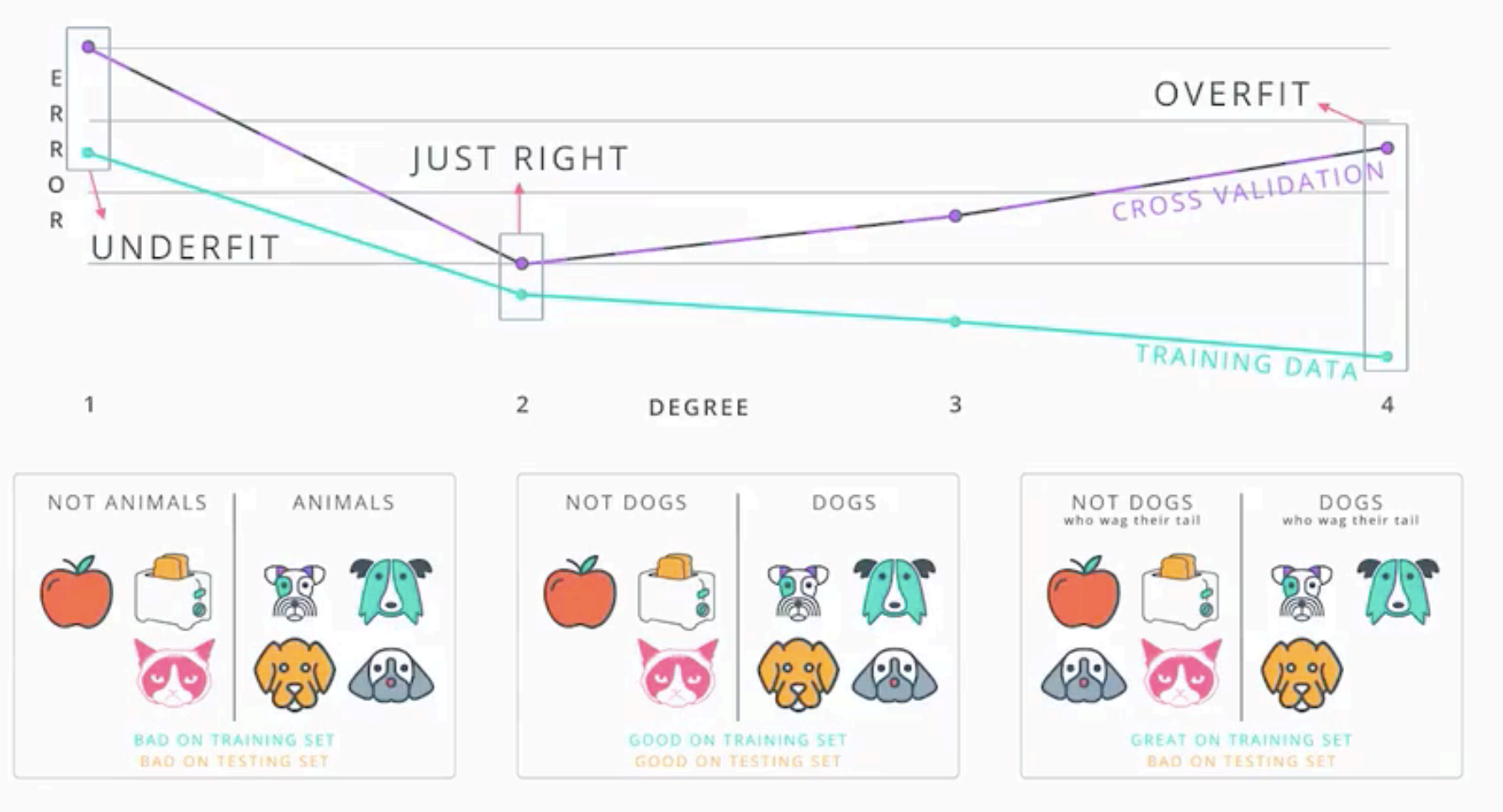
Model Complexity Graph



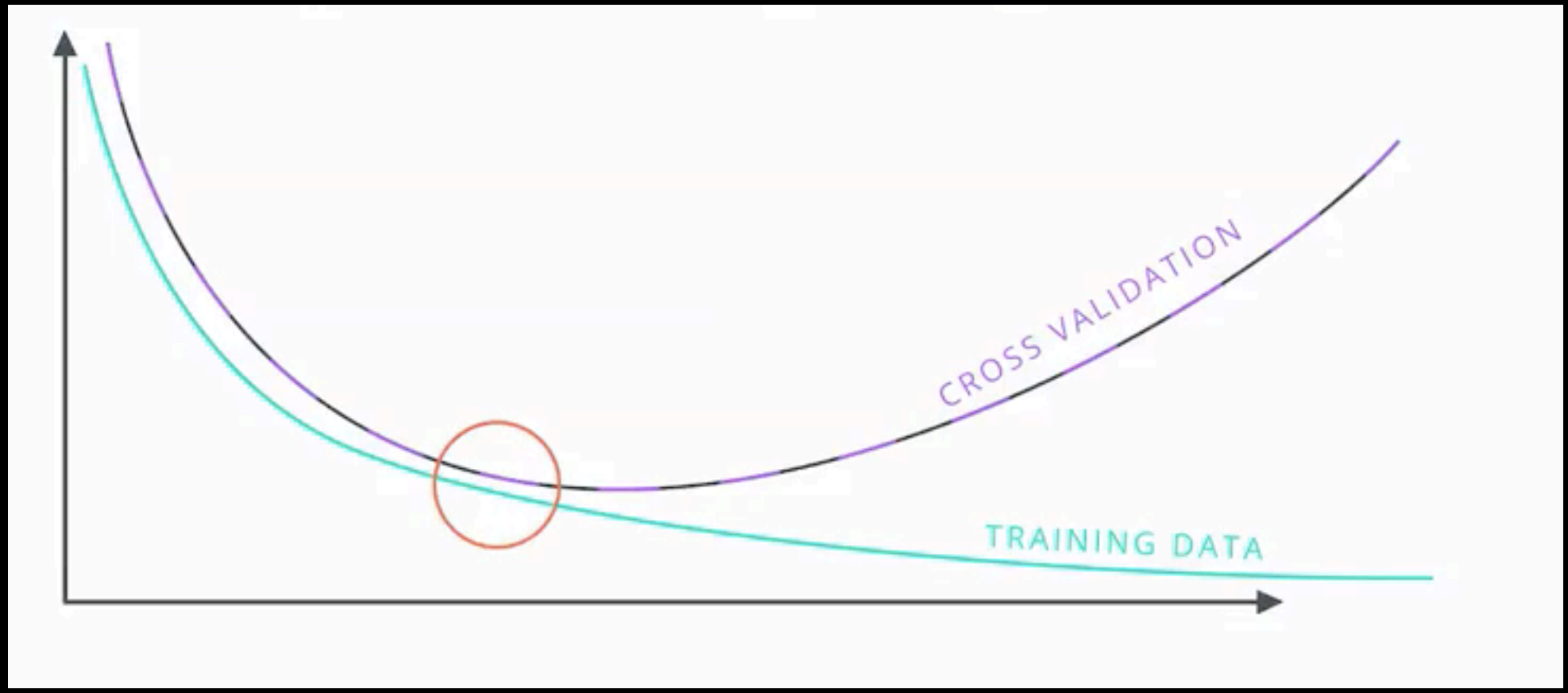
Model Complexity Graph

Cross Validation

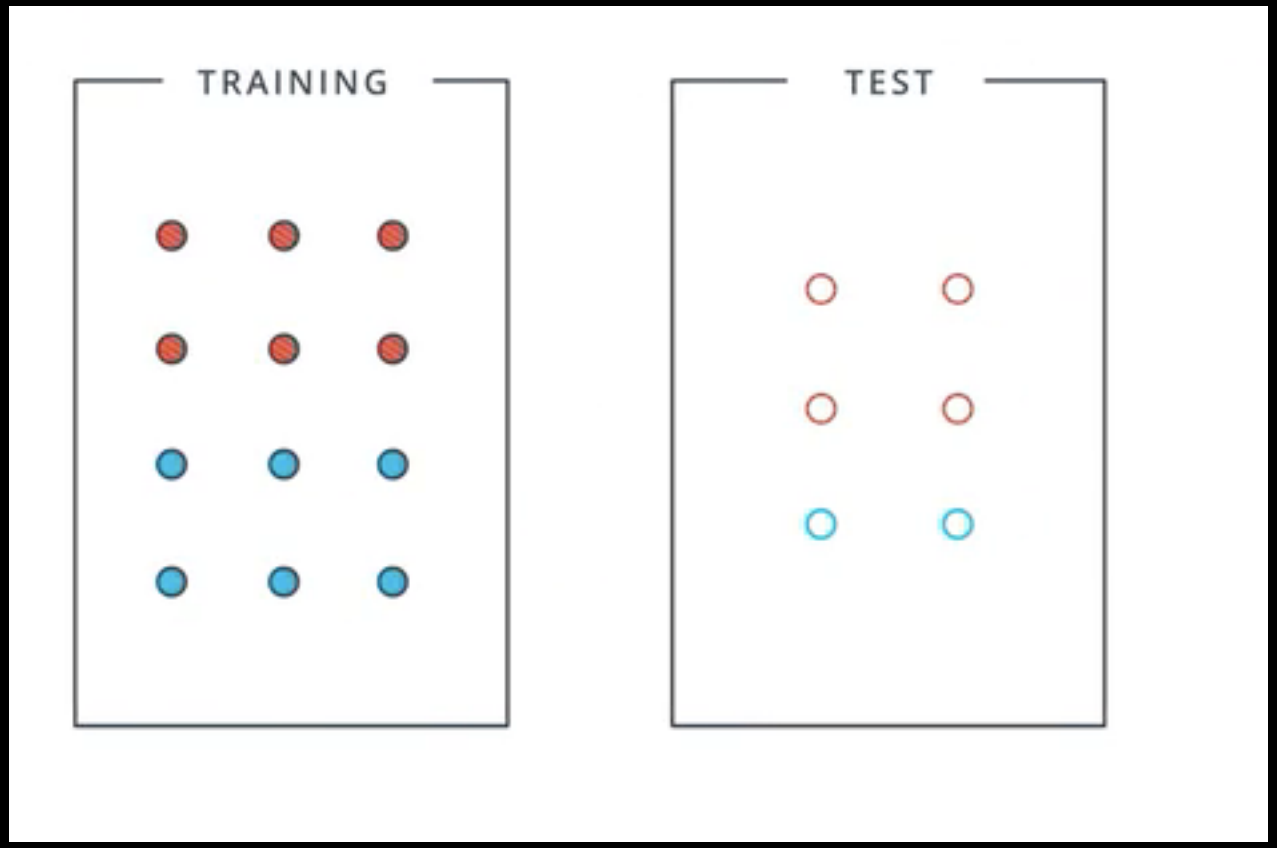
Cross Validation



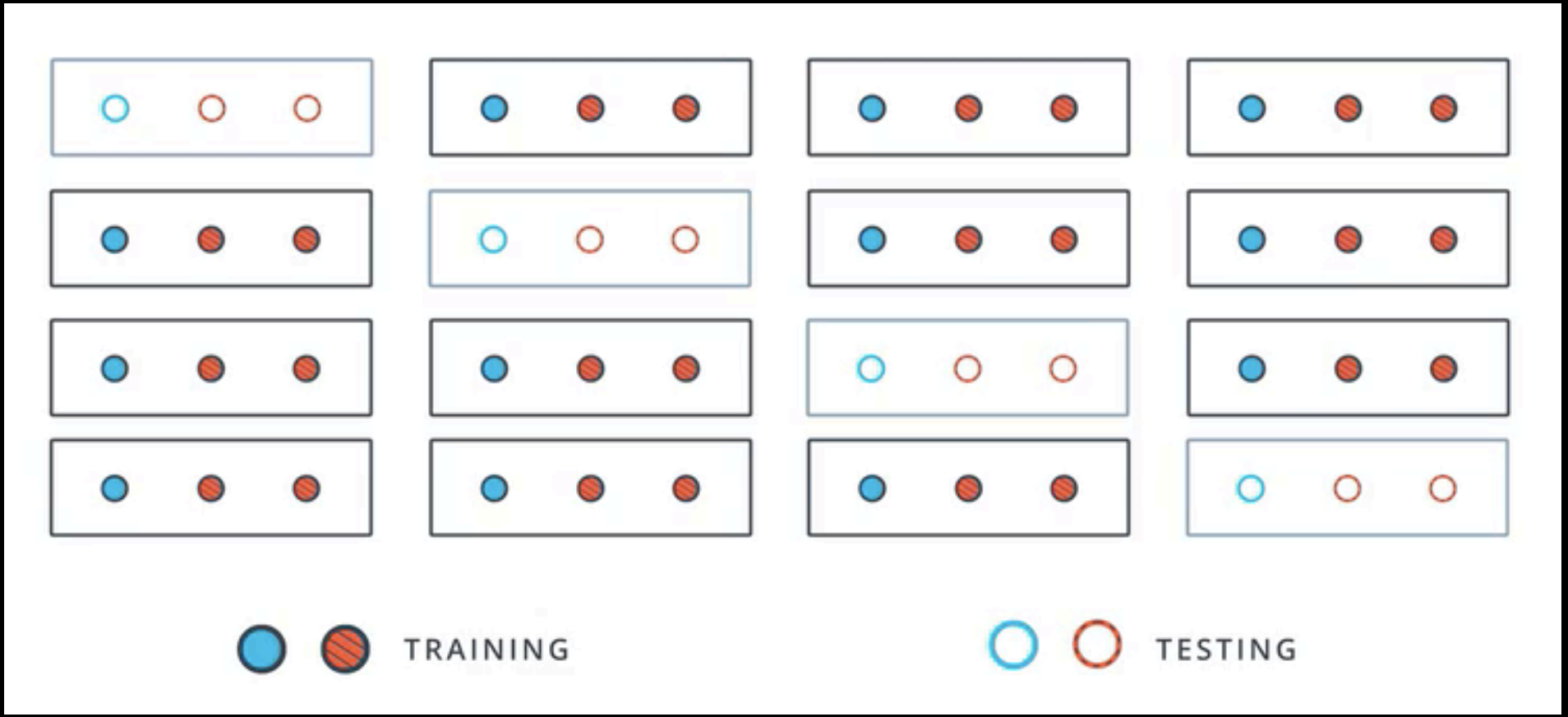
Cross Validation



K-Fold Cross Validation



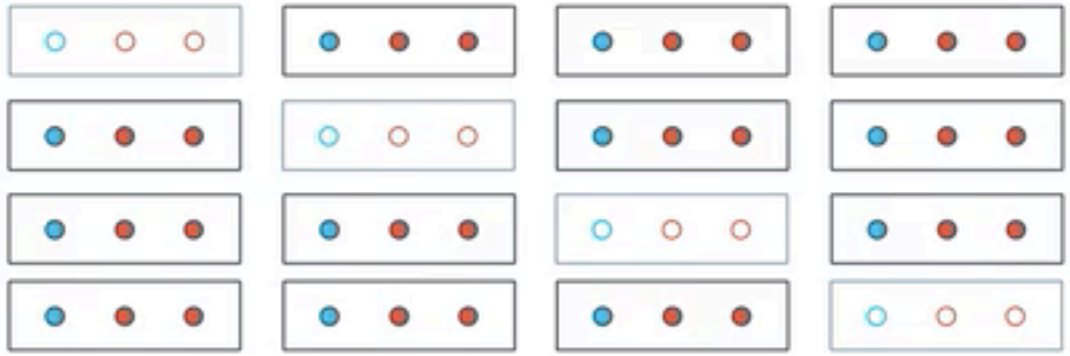
K-Fold Cross Validation



K-Fold Cross Validation

```
from sklearn.model_selection import KFold
kf = KFold(12, 3)

for train_indices, test_indices in kf:
    print train_indices, test_indices
```

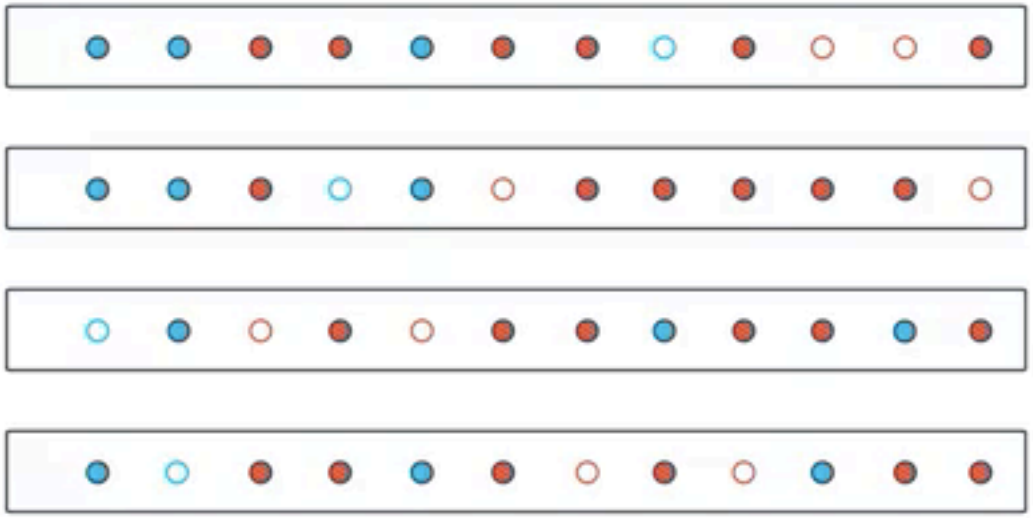
```
[ 3 4 4 6 7 8 9 10 11 ] [ 0 1 2 ]
[ 0 1 2 6 7 8 9 10 11 ] [ 3 4 5 ]
[ 0 1 2 3 4 5 9 10 11 ] [ 6 7 8 ]
[ 0 1 2 3 4 5 6 4  8 ] [ 9 10 11 ]
```



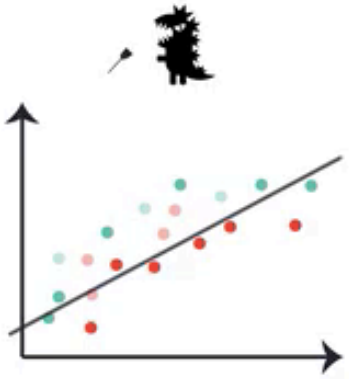
K-Fold Cross Validation

```
from sklearn.model_selection import KFold
kf = KFold(12, 3, shuffle = True)

for train_indices, test_indices in kf:
    print train_indices, test_indices
```

```
[ 0  1  2  3  4  5  6  8 11 ]  [7  9 10]
[ 0  1  2  4  6  7  8  9 10 ]  [3  5 11]
[ 1  3  5  6  7  8  9 10 11 ]  [0  2  4]
[ 0  2  3  4  5  7  9 10 11 ]  [1  6  8]
```



Learning Curves Summary



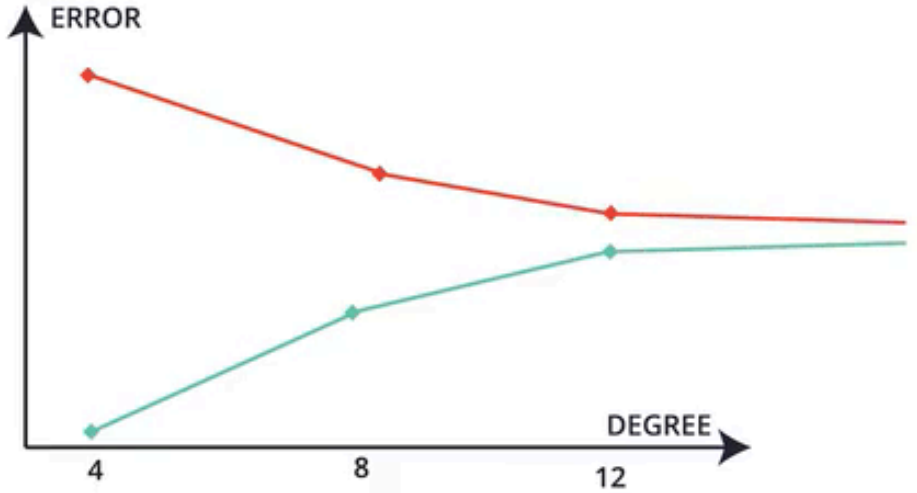
HIGH BIAS
DEGREE = 1

LEARNING CURVES

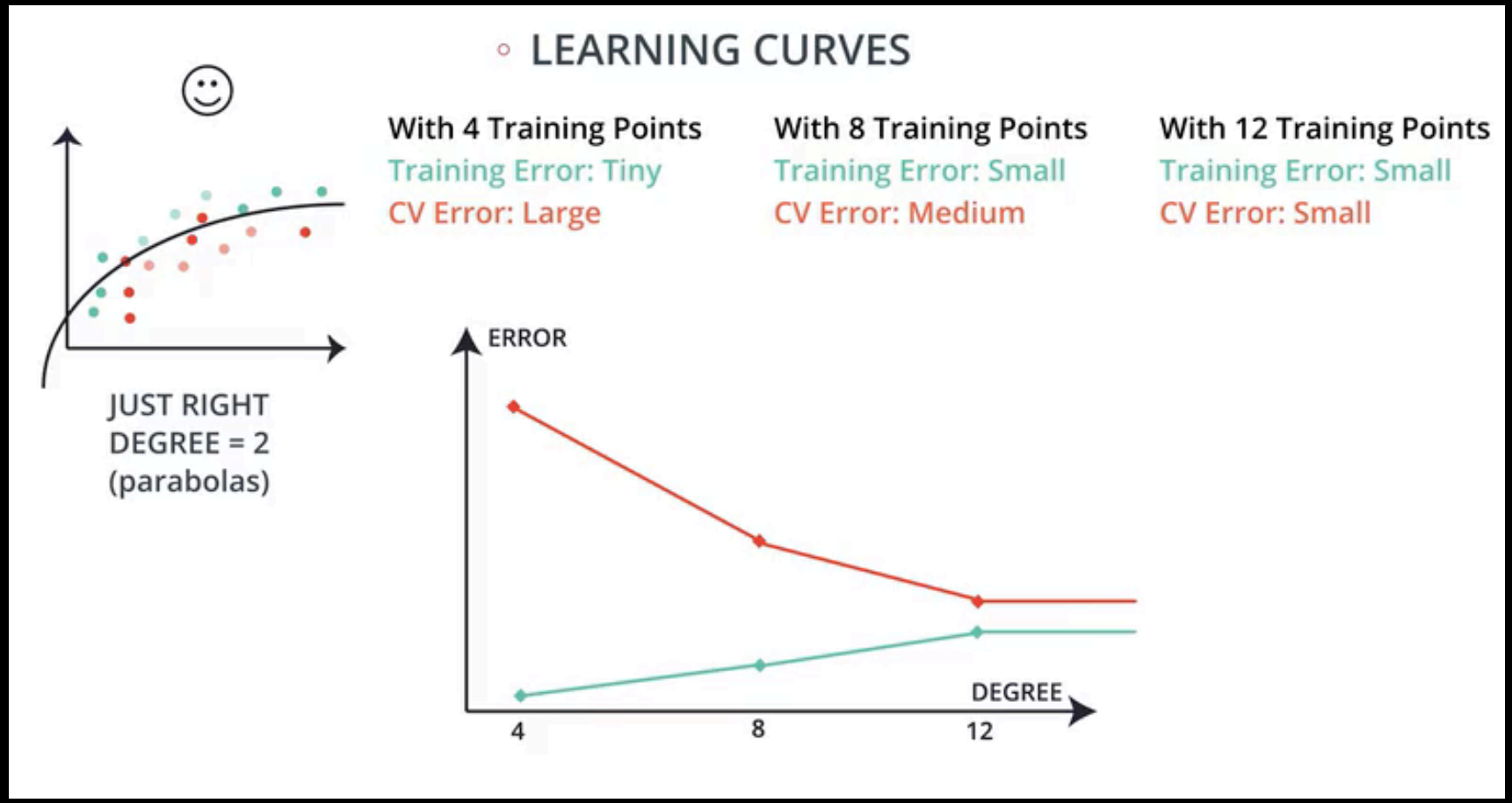
With 4 Training Points
Training Error: Tiny
CV Error: Large

With 8 Training Points
Training Error: Small
CV Error: Medium

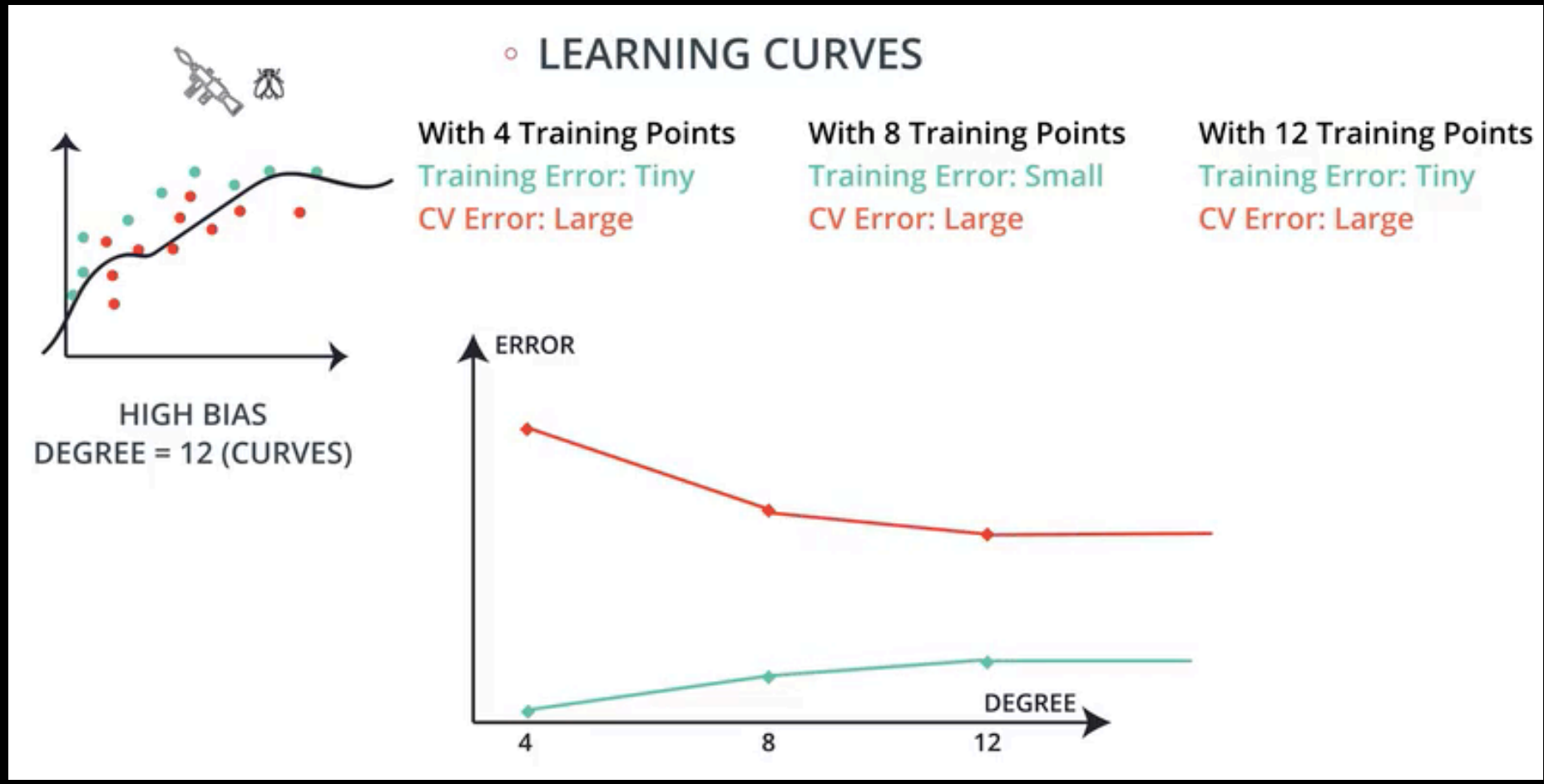
With 12 Training Points
Training Error: Small
CV Error: Medium



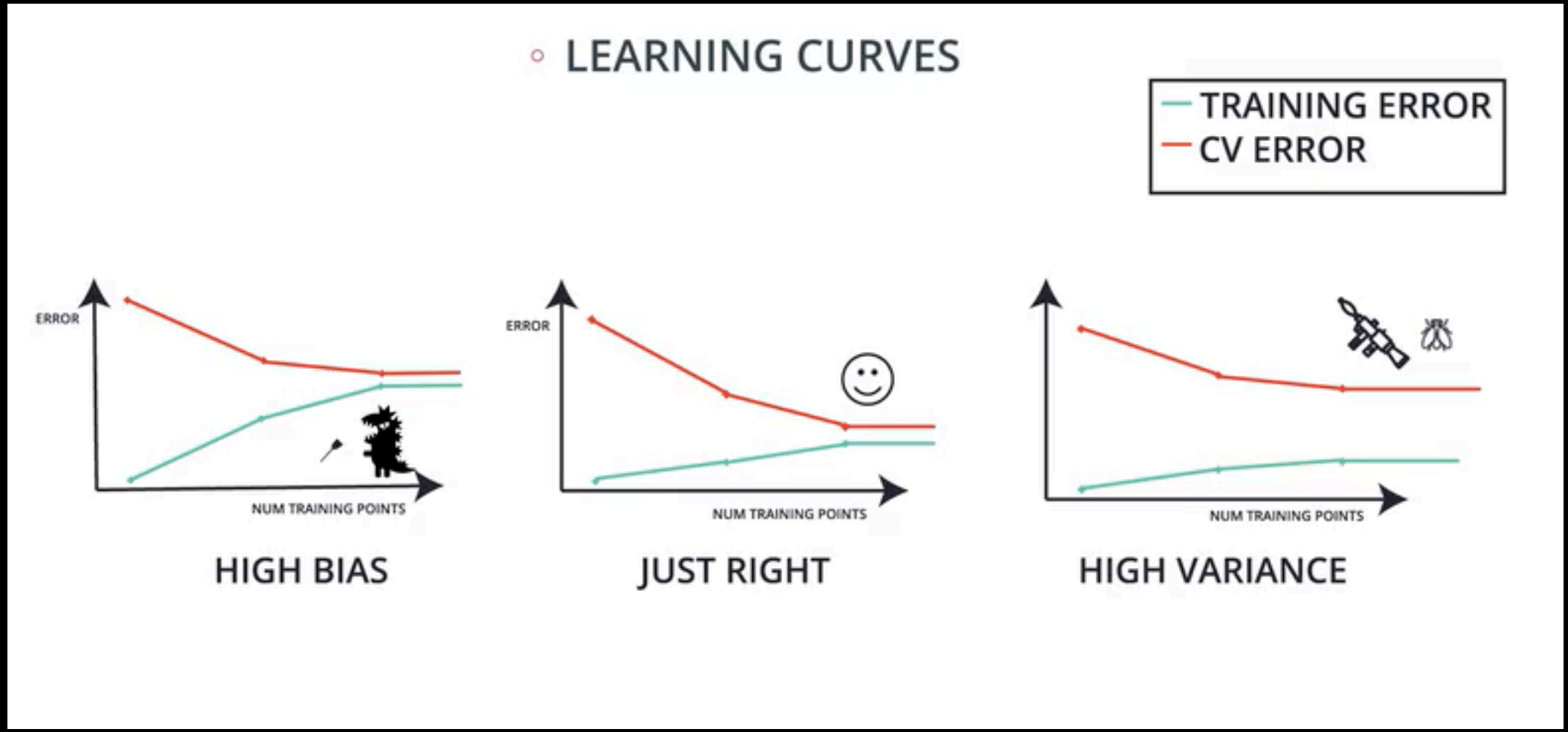
Learning Curves Summary



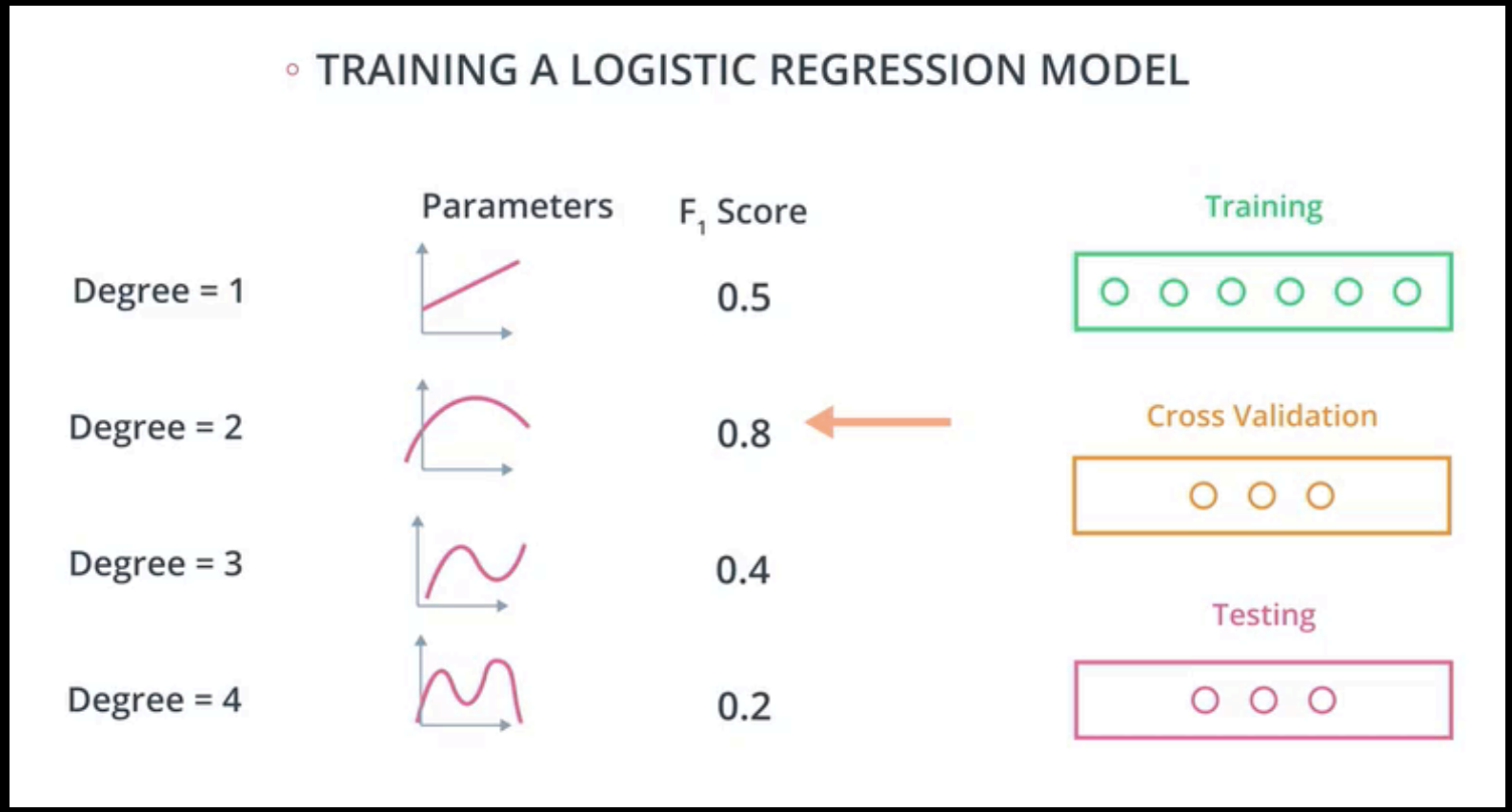
Learning Curves Summary



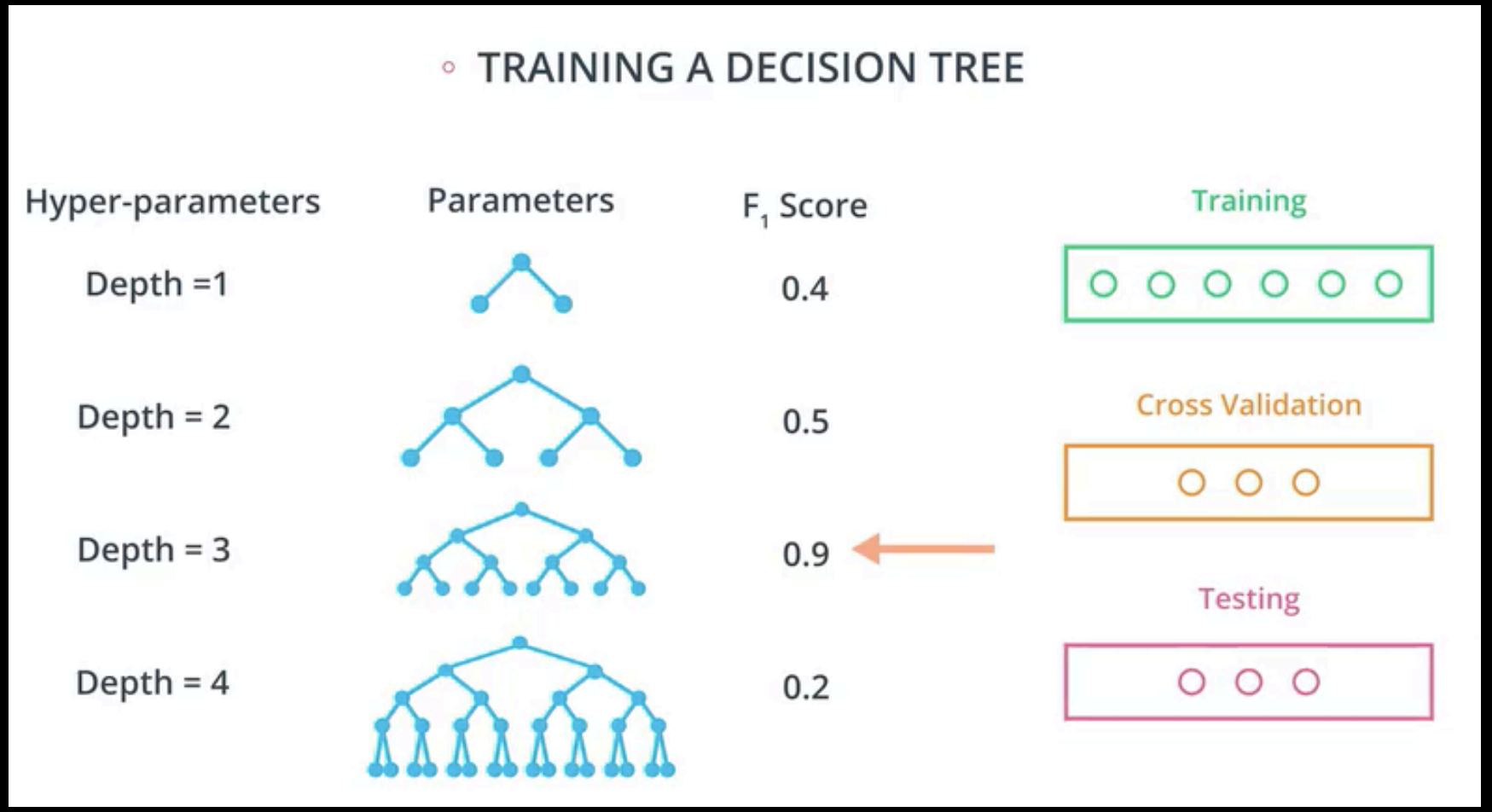
Learning Curves Summary



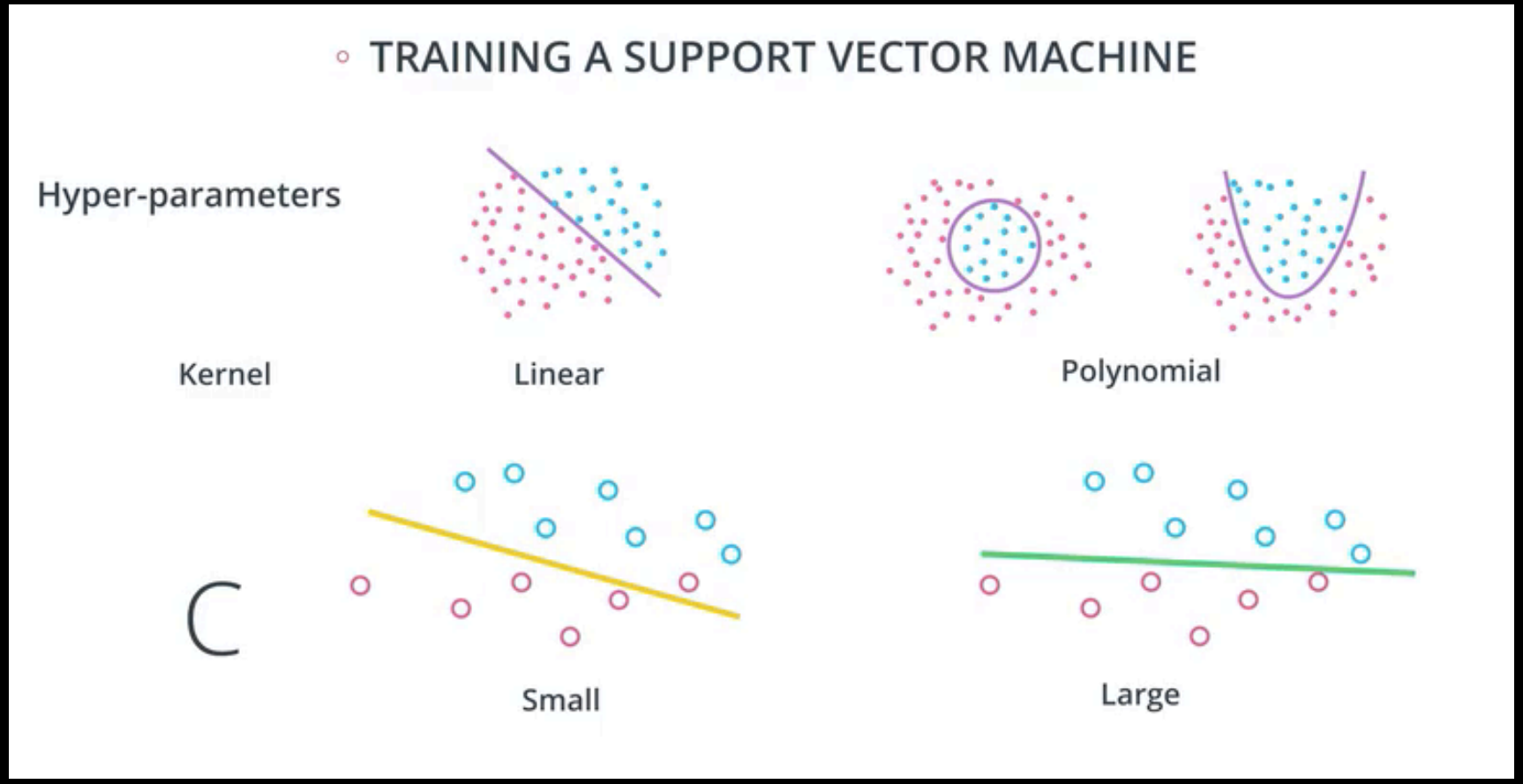
Grid Search


TRAINING A LOGISTIC REGRESSION MODEL
Parameters
F_1 Score
Degree = 1
0.5
Degree = 2
0.8
Degree = 3
0.4
Degree = 4
0.2
Training
Cross Validation
Testing

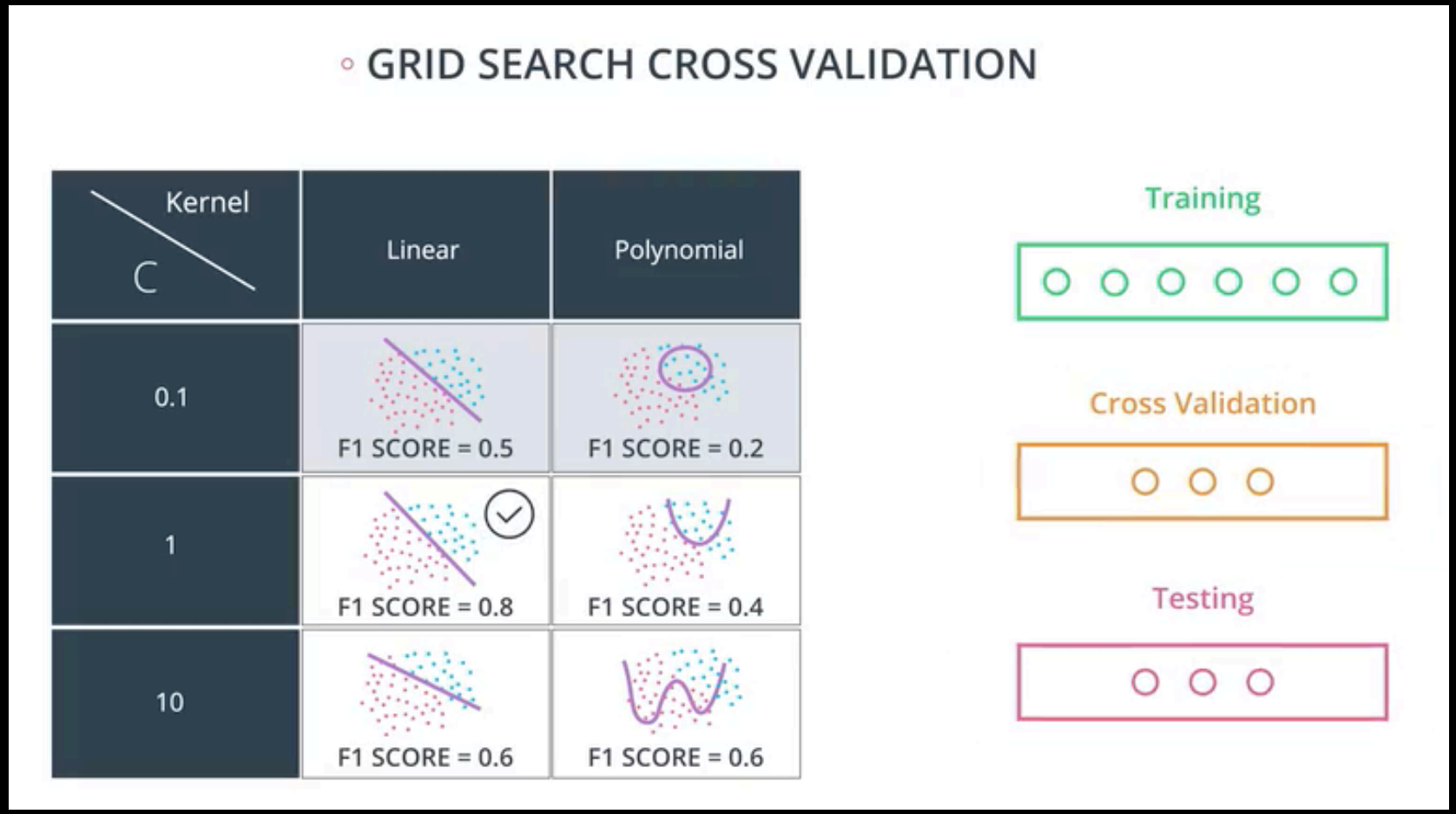
Grid Search


TRAINING A DECISION TREE
Hyper-parameters
Parameters
F_1 Score
Depth =1
0.4
Depth = 2
0.5
Depth = 3
0.9
Depth = 4
0.2
Training
Cross Validation
Testing

Grid Search


TRAINING A SUPPORT VECTOR MACHINE
Hyper-parameters
Kernel
Linear
Polynomial
C
Small
Large

Grid Search


GRID SEARCH CROSS VALIDATION
Kernel
C
Linear
Polynomial
0.1
F1 SCORE = 0.5
F1 SCORE = 0.2
1
F1 SCORE = 0.8
F1 SCORE = 0.4
10
F1 SCORE = 0.6
F1 SCORE = 0.6
Training
Cross Validation
Testing