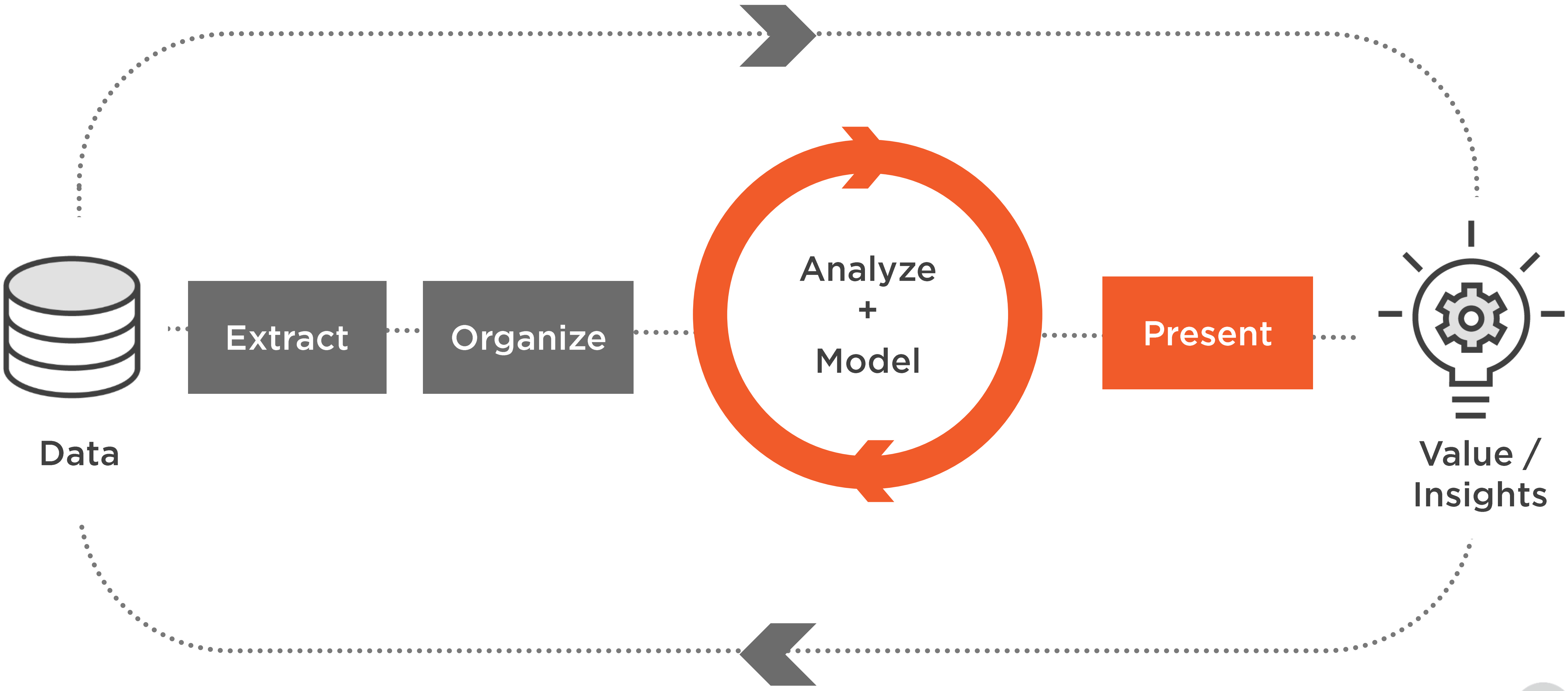
Building and Evaluating Predictive Models – Part 2

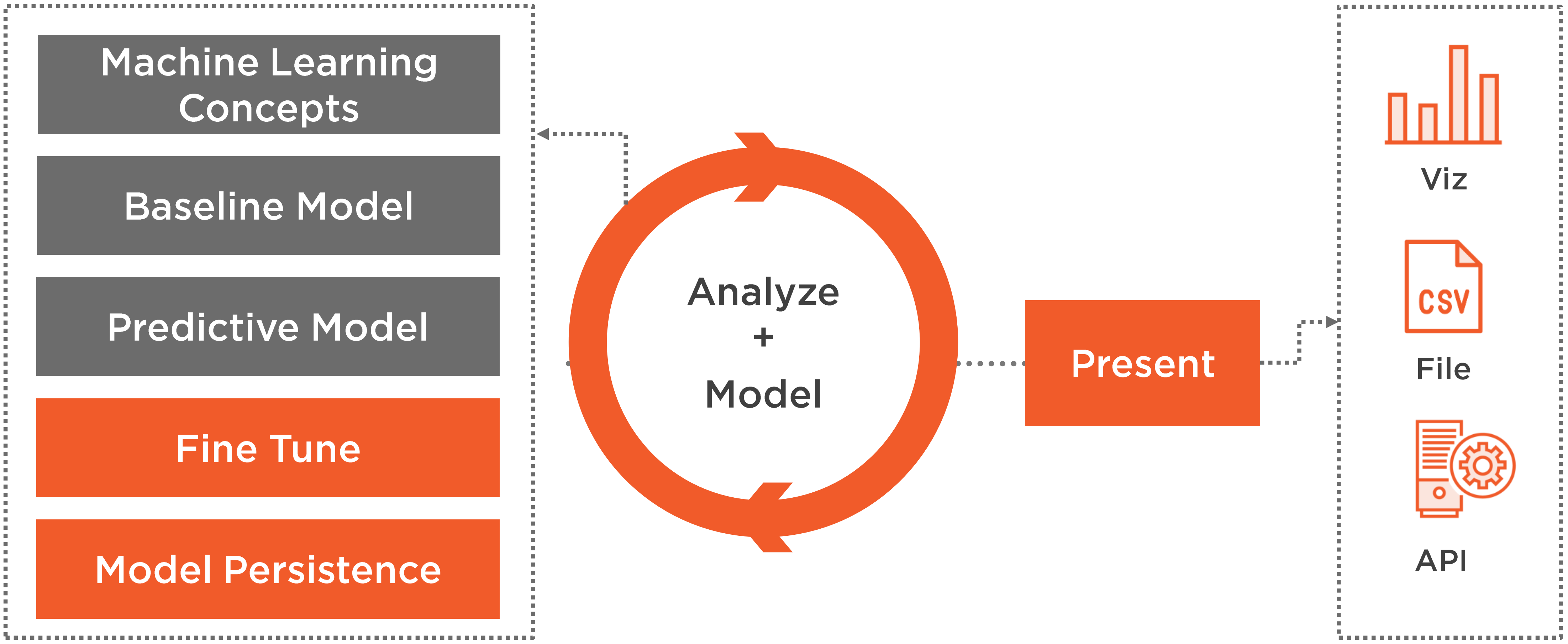
Abhishek Kumar
AUTHOR

@meabhishekkumar

Data Science Project Cycle



Data Science Project Cycle



Machine Learning Concepts

Baseline Model

Predictive Model

Fine Tune

Model Persistence

Analyze + Model

Present

Viz

CSV

File

API

Overview

(Contents)

Model tuning

- Underfitting vs overfitting
- Regularization
- Hyperparameter tuning
- Cross validation

Feature Engineering

- Feature normalization

Model persistence

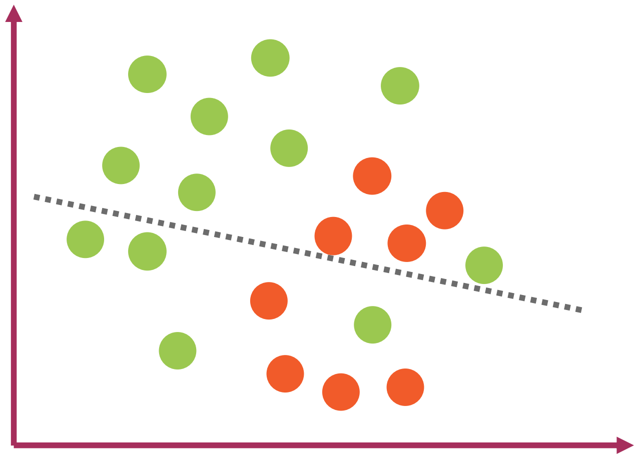
API

Overview (Tools)

Python

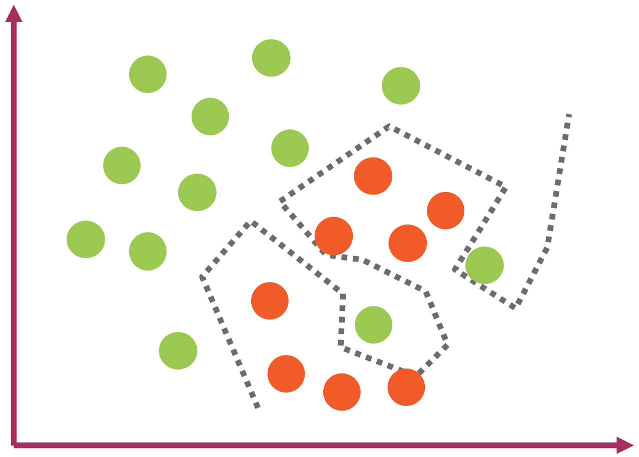
- Numpy
- Pandas
- Scikit-Learn
- Pickle
- Flask

Underfitting vs. Overfitting



Underfitting

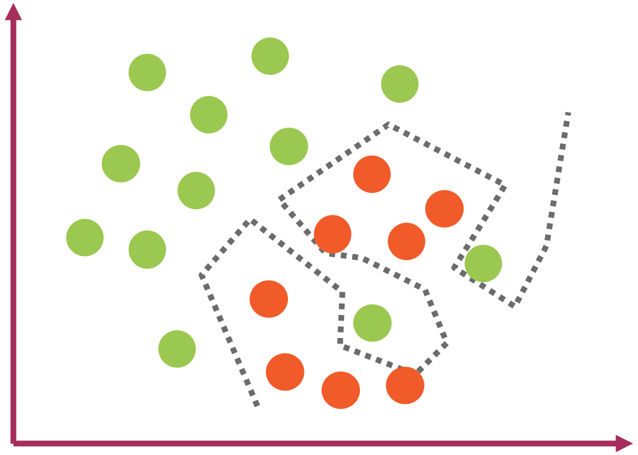
- Can’t learn the pattern in the training data



Overfitting

- Memorize training data
- Poor generalization

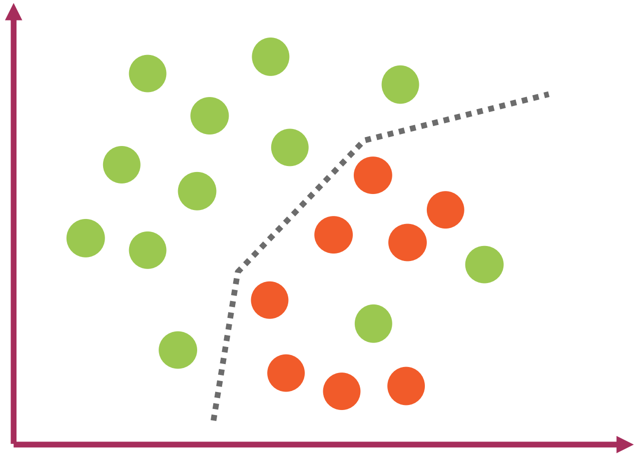
Regularization



Overfitting

Regularization

Reduce model complexity



Balanced

Regularization

Create model

```
model = LogisticRegression(random_state=0)
```

Train model

```
model.fit(X_train, y_train)
```

Score

```
model.score(X_test, y_test)
```

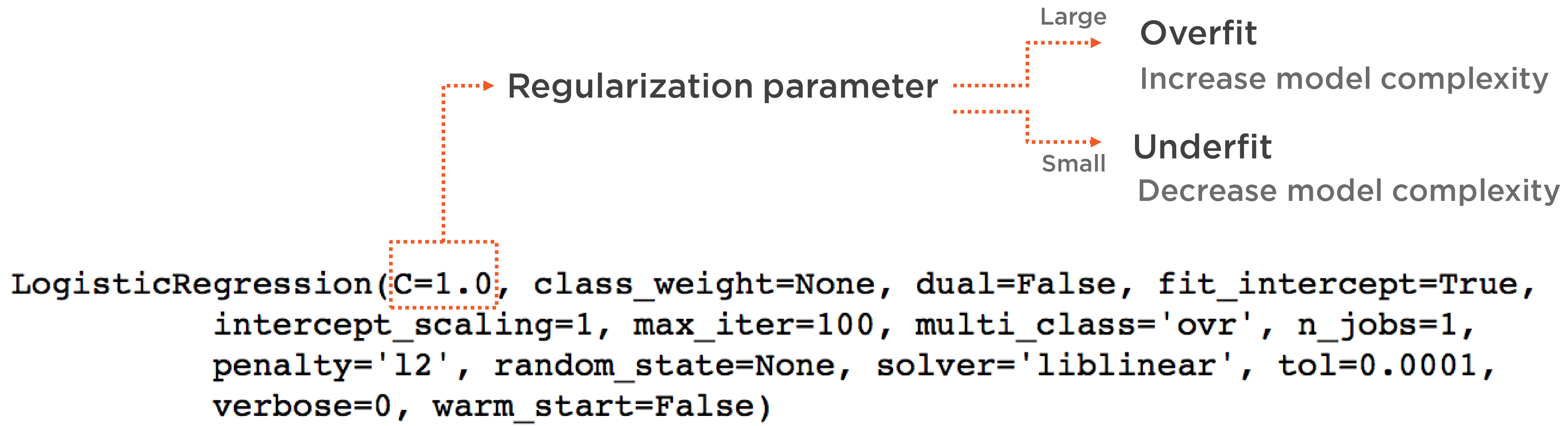
Coefficients (Model parameters)

```
model.coef_
```

Regularization

Create model

model = LogisticRegression(random_state=0)



Regularization parameter

Large

Overfit

Increase model complexity

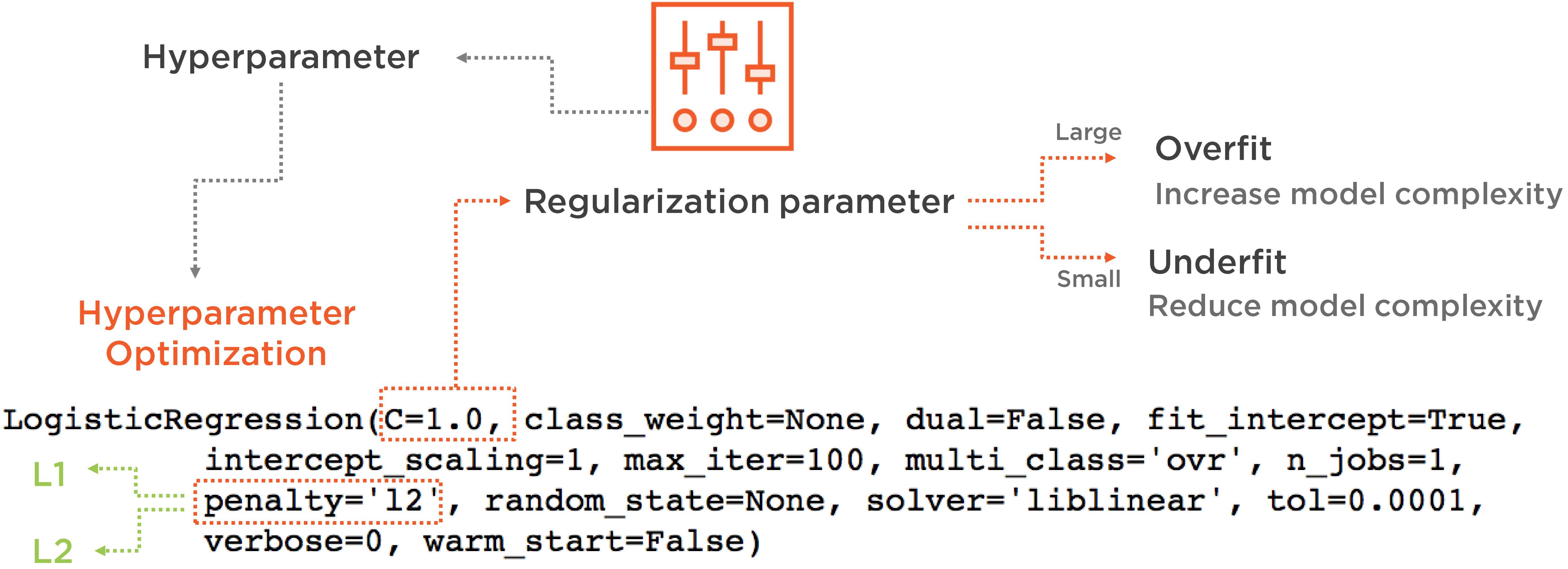
Small

Underfit

Decrease model complexity

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

Regularization



Hyperparameter

Hyperparameter Optimization

Regularization parameter

Large

Overfit

Increase model complexity

Small

Underfit

Reduce model complexity

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

Hyperparameter Optimization : GridSearch

Model (A)

a1	a2	a3

Model (A, B)

	a1	a2	a3
b1			
b2			
b3			
b4			

Create grid with different combinations

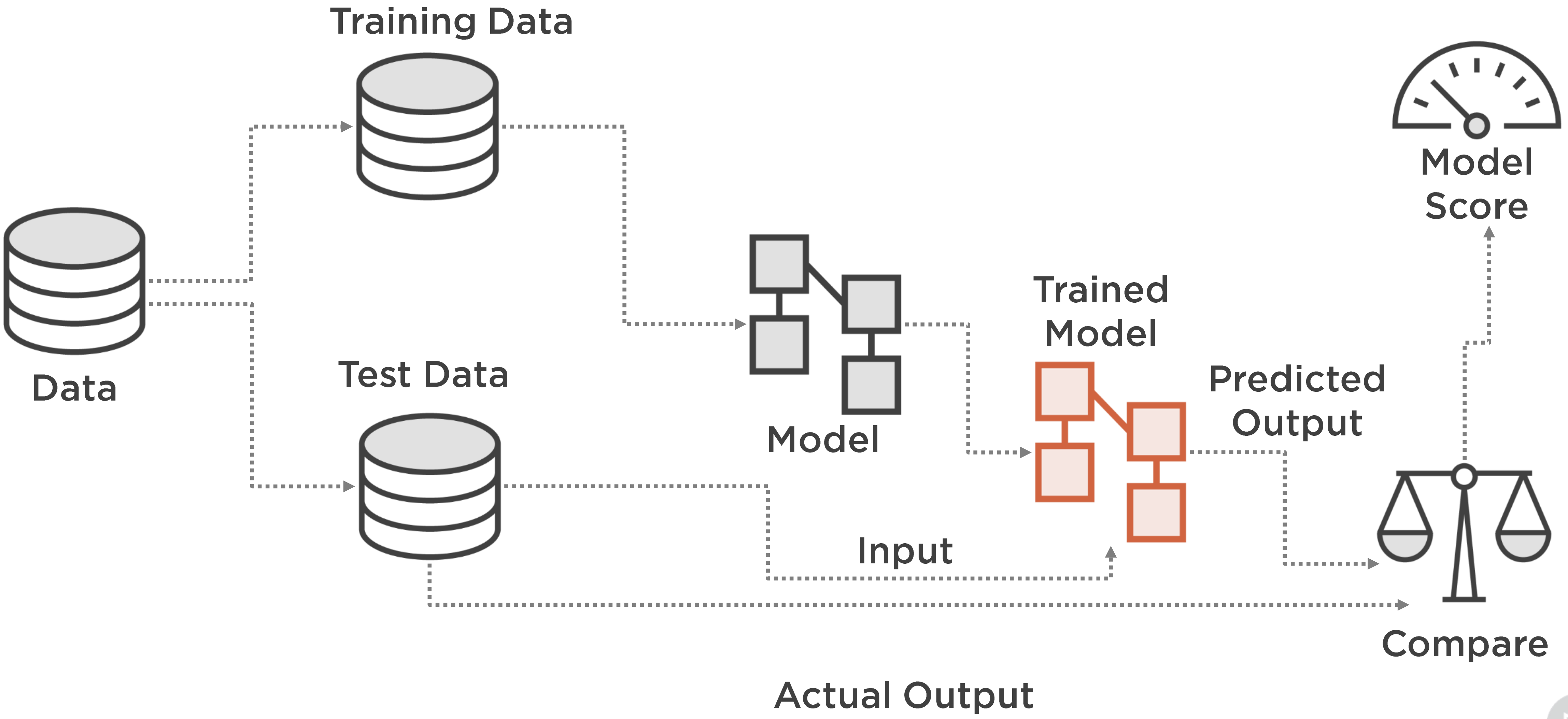
→

Evaluate each combination

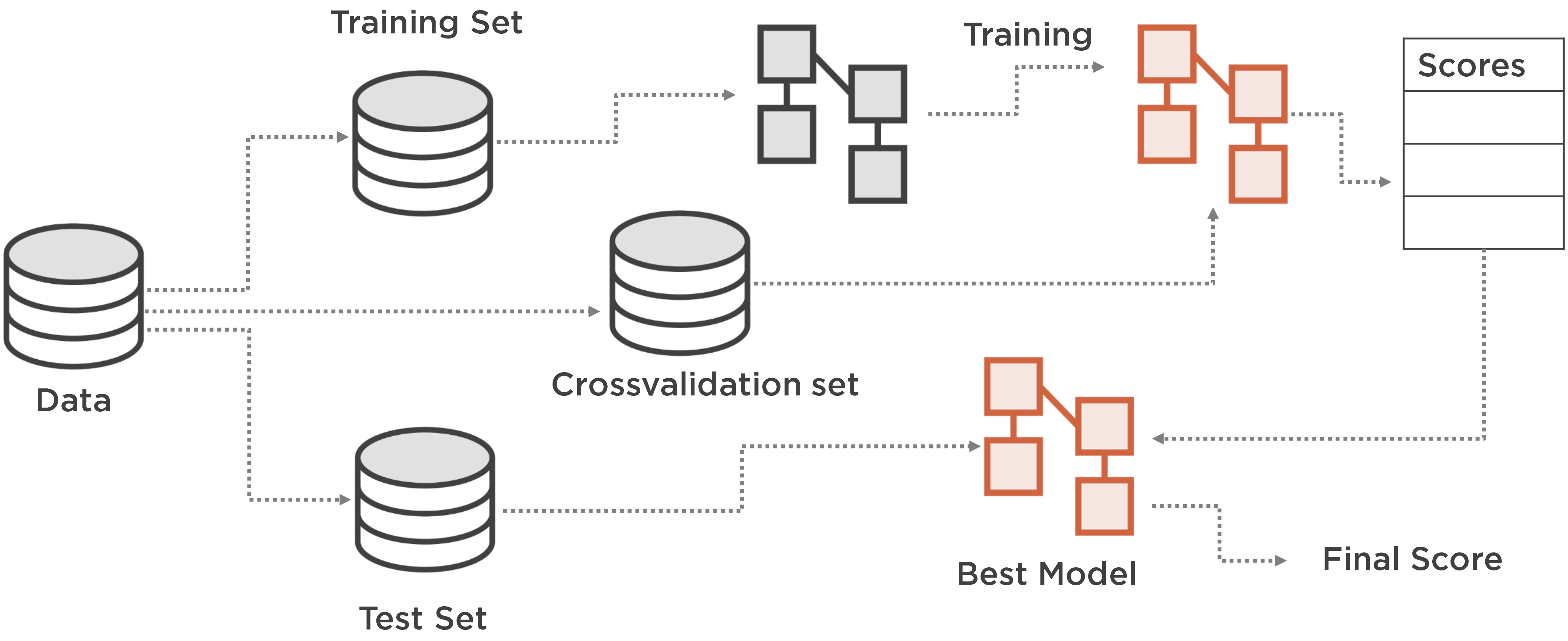
→

Select combination with best model performance

Train-test Split

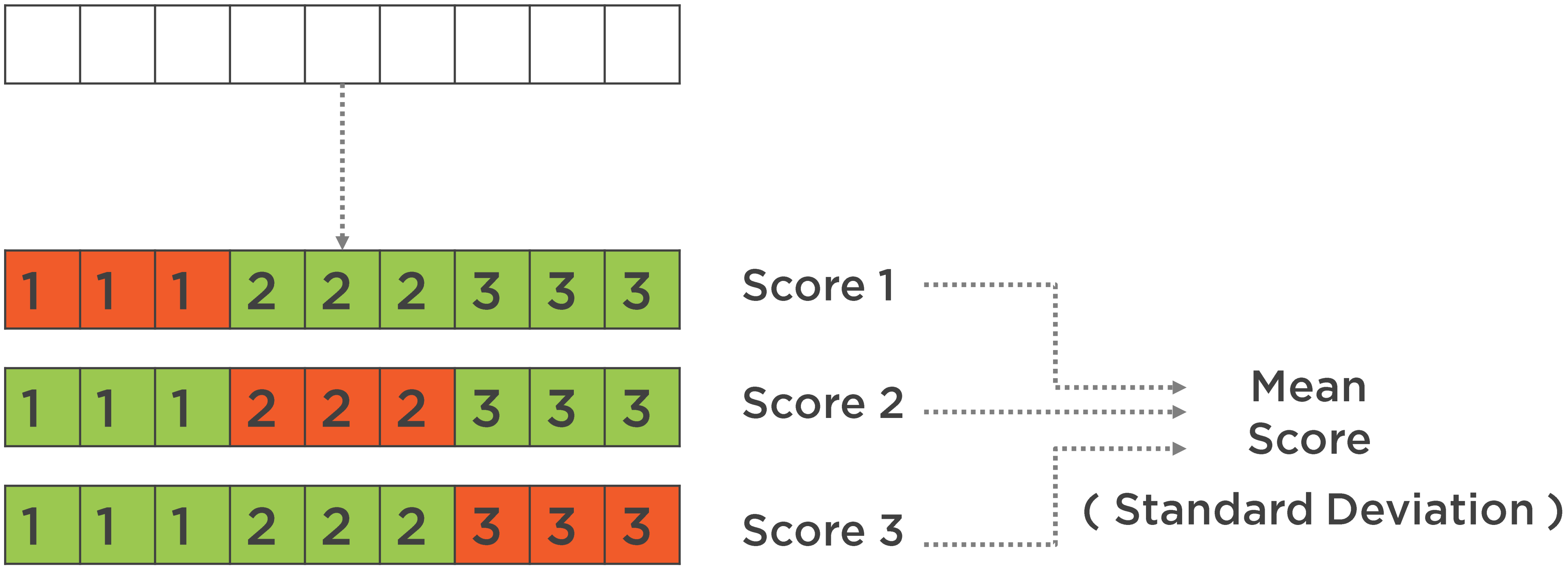


Cross-validation



K-Fold Cross-validation

K = 3, 3-Fold



1 1 1 2 2 2 3 3 3

1 1 1 2 2 2 3 3 3

1 1 1 2 2 2 3 3 3

Score 1

Score 2

Score 3

Mean Score

(Standard Deviation)

Demo

Hyperparameter optimization using GridSearchCV

Demo

Making third Kaggle submission

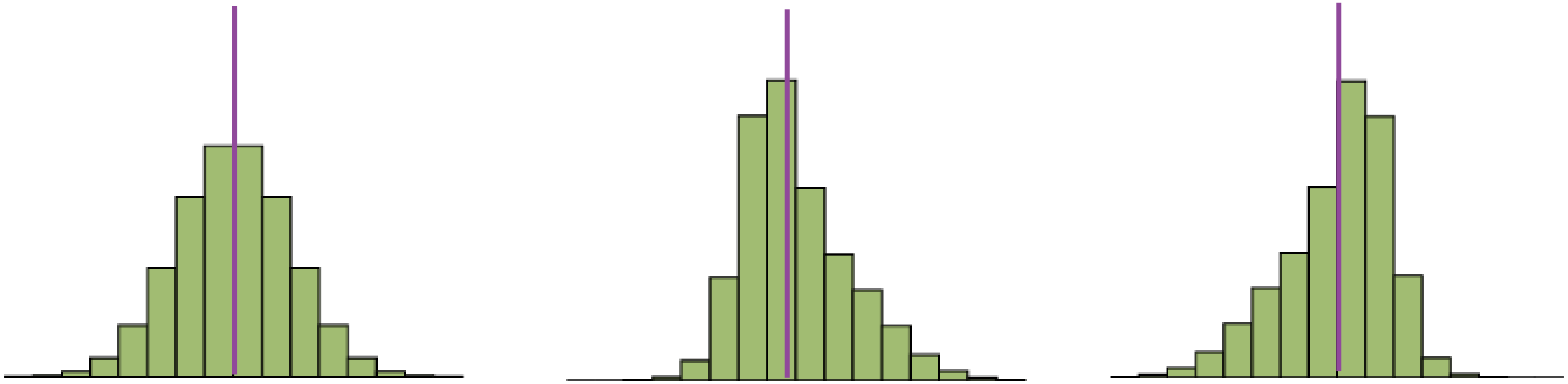
Feature Normalization

Age	Fare	FamilySize
..	..	..
..	..	..
..	..	..

0.4 to 80	0 to 512	1 - 11	
0 to 1	0 to 1	0 to 1	Scale Type 1
-1 to 1	-1 to 1	-1 to 1	Scale Type 2

Feature Standardization

0.4 to 80	0 to 512	1 - 11
Age	**Fare**	**FamilySize**
..	..	..
..	..	..
..	..	..

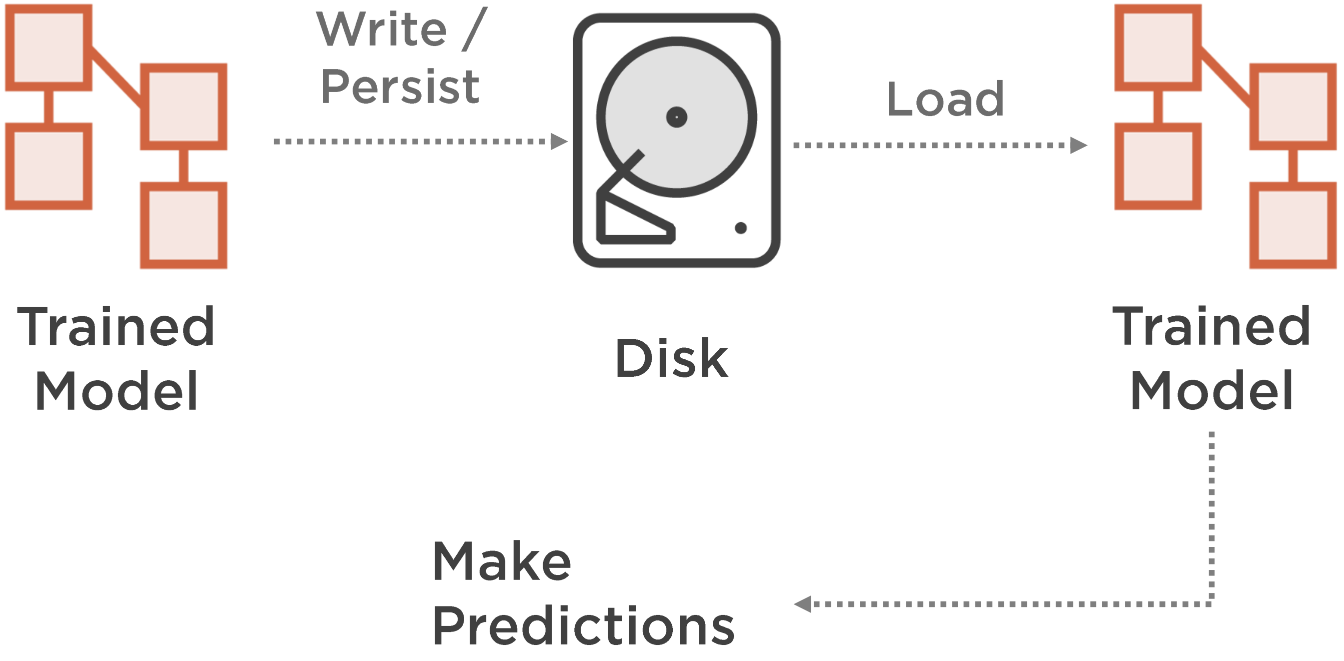


Mean = 0.0
Variance = 1.0

Demo

Feature normalization and standardization using Scikit-Learn

Model Persistence



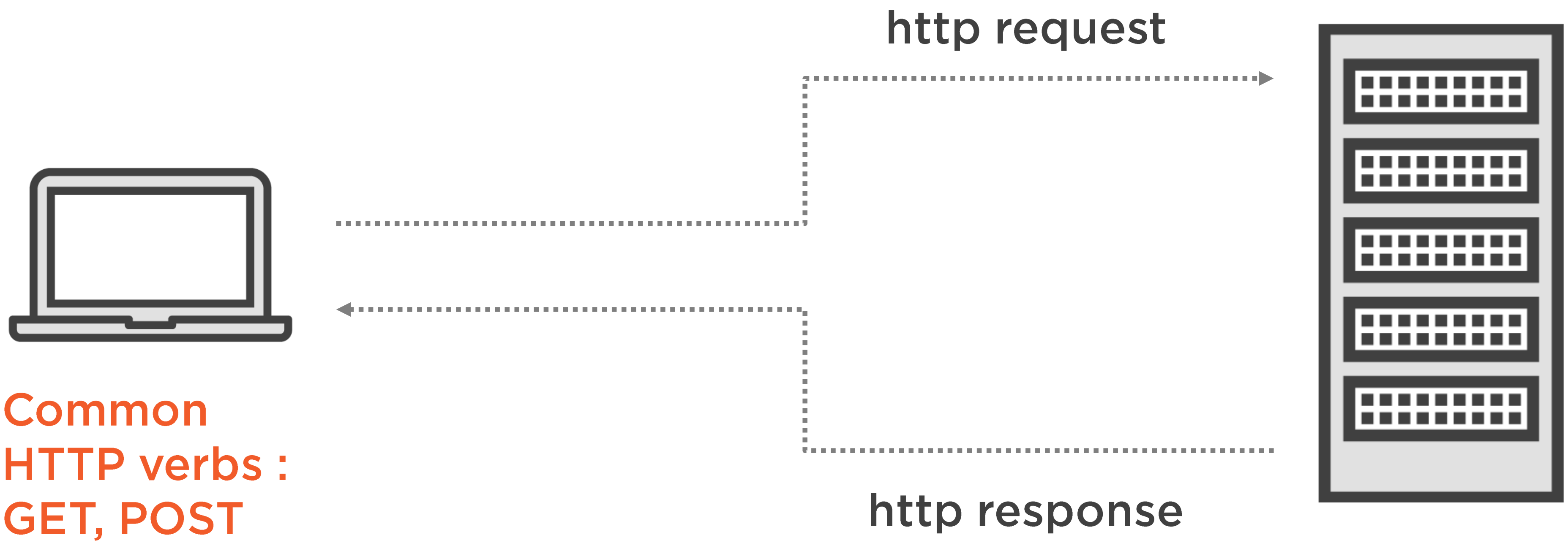
- No retraining required
- Share trained model
- Can be used to create machine learning APIs

Demo

Model persistence using Pickle

Machine Learning API Development

REST API

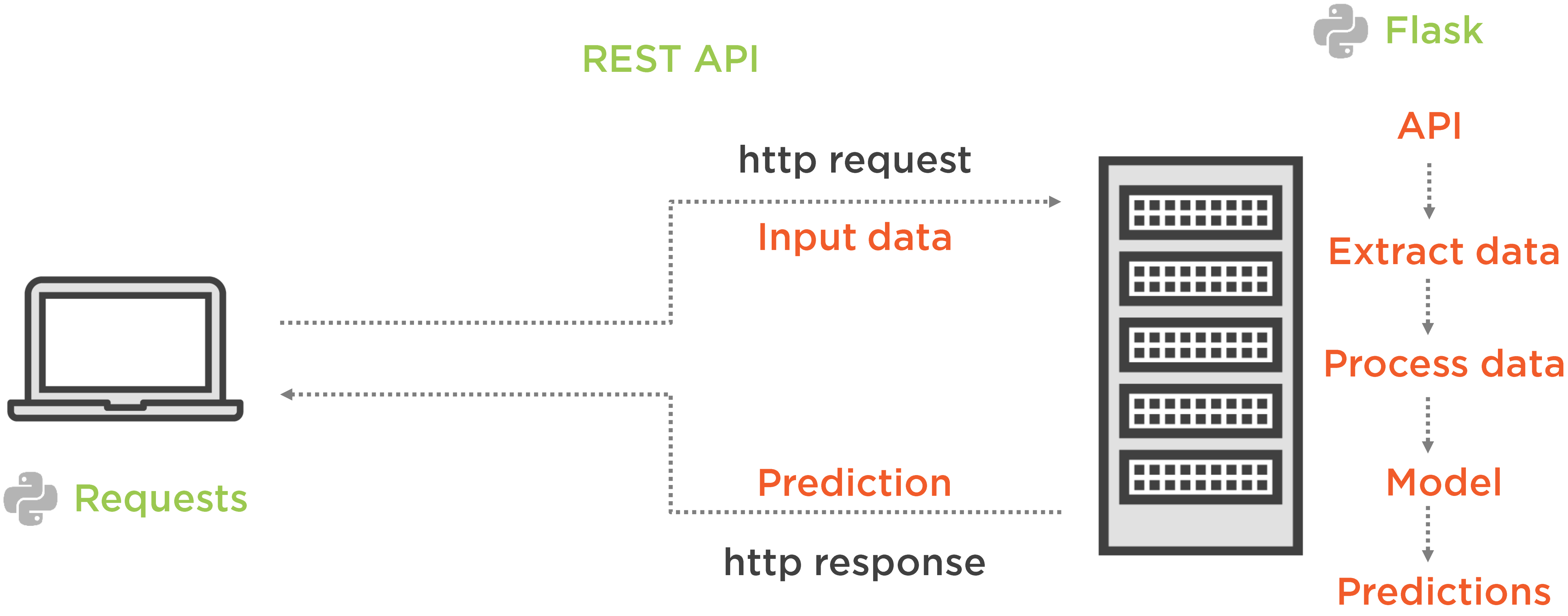

http request

http response

Common HTTP verbs : GET, POST

Requests

Machine Learning API Development



REST API

http request

Input data

Prediction

http response

Requests

Flask

API

Extract data

Process data

Model

Predictions

Demo

Hello world API using Flask

Demo

Machine Learning API using Flask

Demo

Committing changes to git

Summary

Model tuning

- Underfitting vs overfitting
- Hyperparameter tuning

Feature normalization

Model persistence

Machine learning API development

Where to Go from Here?

- Datasets
- Machine learning algorithms
- Pipelines, API
- Community
- Competitions