CORNELL CENTER for
SOCIAL SCIENCES

Accelerates, enhances, and amplifies social science research at Cornell.

Supervised Learning in Python

Angel Hwang
Senior Data Science Fellow
hh695@cornell.edu
CCSS-ResearchSupport@cornell.edu

CCSS Research Support Code of Conduct

The Cornell Center for Social Sciences provides a welcoming environment for everyone embracing all backgrounds or identities. All instructors and attendees agree to abide by our community norms. We encourage the following behaviors in our workshops:

- Respect differing viewpoints and ideas
- Share your own perspectives and ask any questions
- Accept constructive criticism
- Use welcoming and inclusive language
- Show courtesy and respect for all instructors and attendees

If you believe that an instructor or attendee has violated the code of conduct, please report the violation to CCSS-ResearchSupport@cornell.edu. We take all reported incidents seriously.

Land Acknowledgement

Cornell University is located on the traditional homelands of the Gayogo̱hó꞉nǫ' (the Cayuga Nation). The Gayogo̱hó꞉nǫ' are members of the Haudenosaunee Confederacy, an alliance of six sovereign Nations with a historic and contemporary presence on this land. The Confederacy precedes the establishment of Cornell University, New York state, and the United States of America. We acknowledge the painful history of Gayogo̱hó꞉nǫ' dispossession, and honor the ongoing connection of Gayogo̱hó꞉nǫ' people, past and present, to these lands and waters.

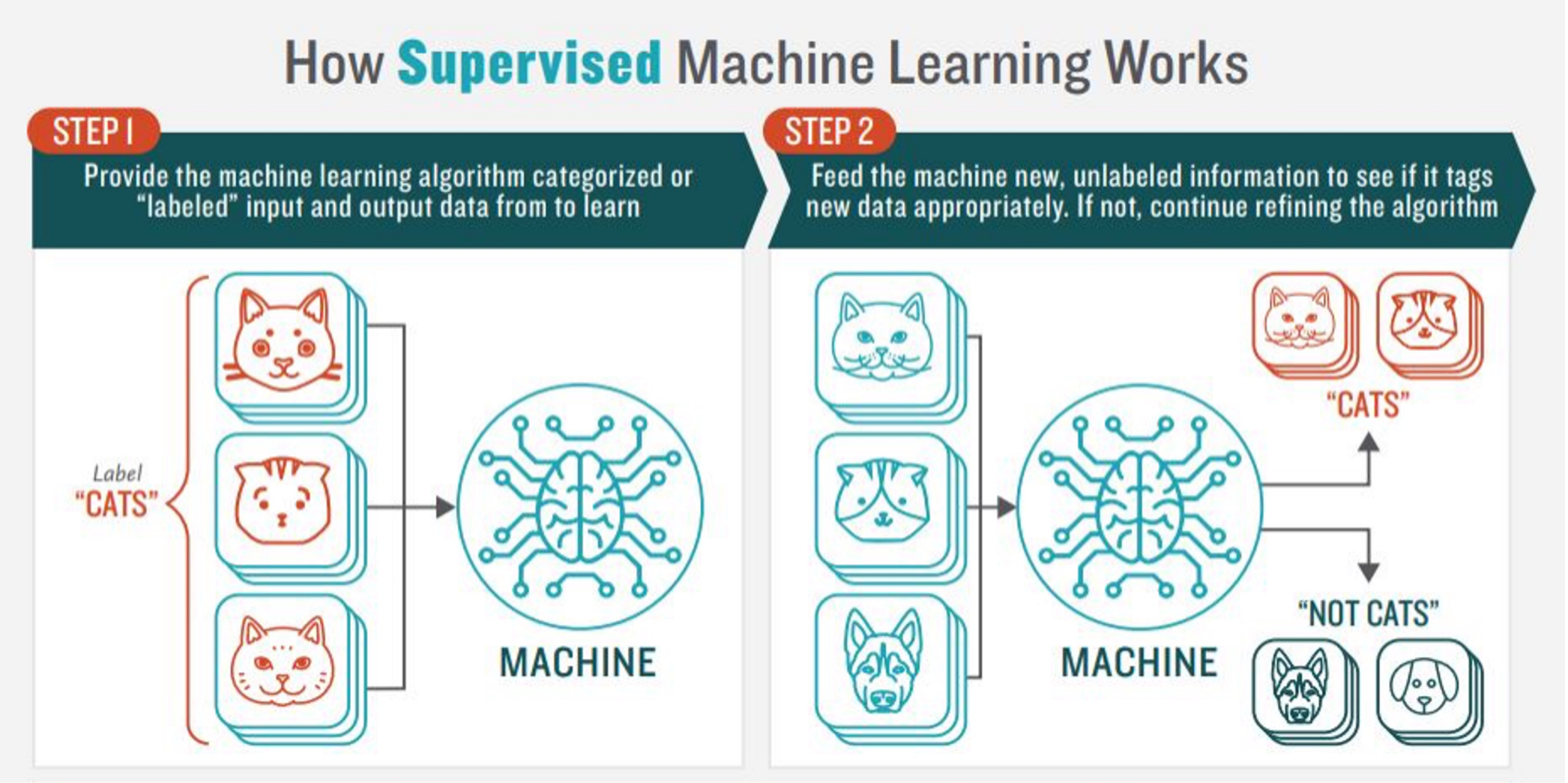
Here are additional links for more on the history of Cornell's violent, colonial formation, the movement to return native lands, and about the AIISP program at Cornell.

Consider donating to the Gayogo̱hó꞉nǫ' sovereignty initiative here.

Outline

- **Introduction**
- Core Concepts
- Mini Case Study
- Common Challenges
- Conclusion (The Supervised Learning Process)

What is Supervised Learning?



Source: A Quick Guide to How Machine Learns

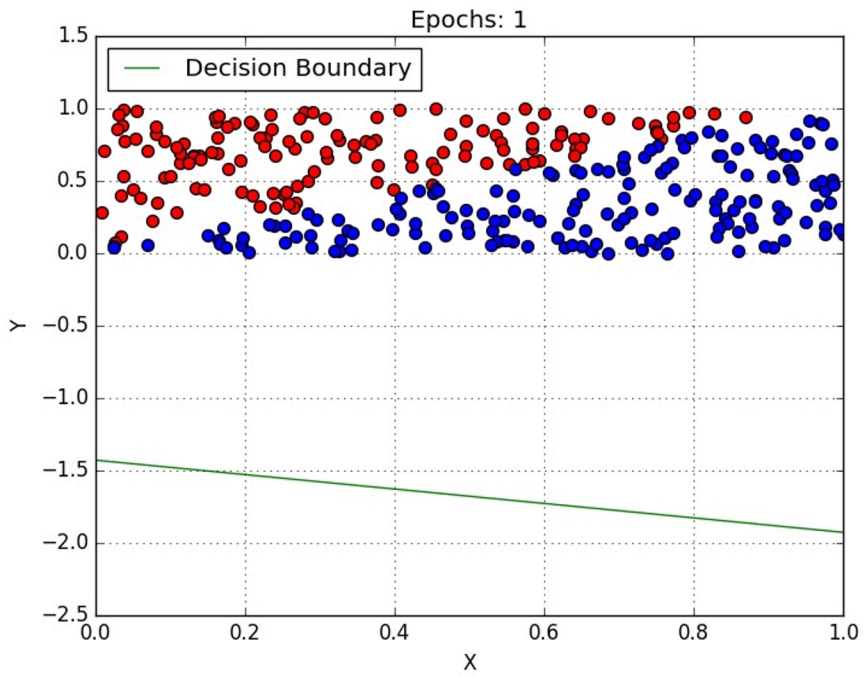
What is Supervised Learning?

- Research questions can be categorized into *classification* tasks and *regression* tasks:
  - *Classification*: the labels are discrete
    - Binary: 0/1, Yes/No
    - Multi-class: Cat/Dog/Tiger, Digit(0,1,..,9)
  - *Regression*: the labels are continuous
    - Wage, housing prices, etc.

What is Supervised Learning?

Classification:

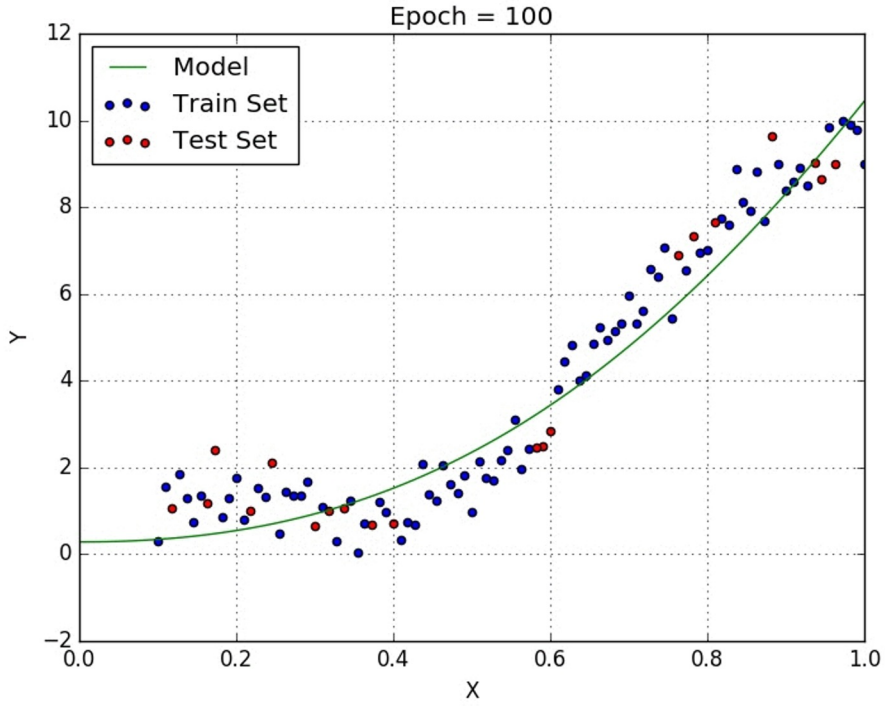
Y: [Cat, Not Cat]. X: Size



Source: Blog post by Davi Frossard

Regression

Y: Housing Prices. X: Garage Size



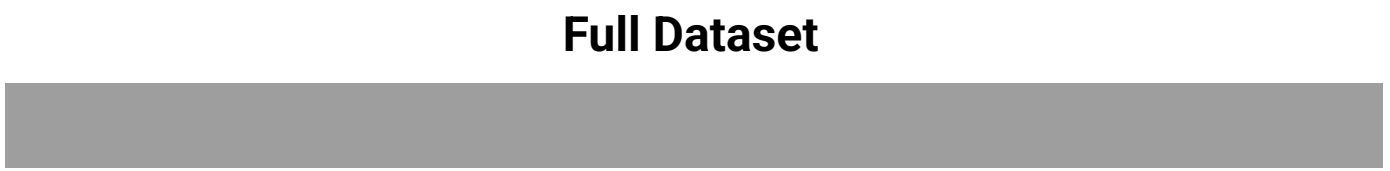
Source: Blog post by Davi Frossard

Outline

- Introduction
- **Core Concepts**
- Mini Case Study
- Common Challenges
- Conclusion (The Supervised Learning Process)

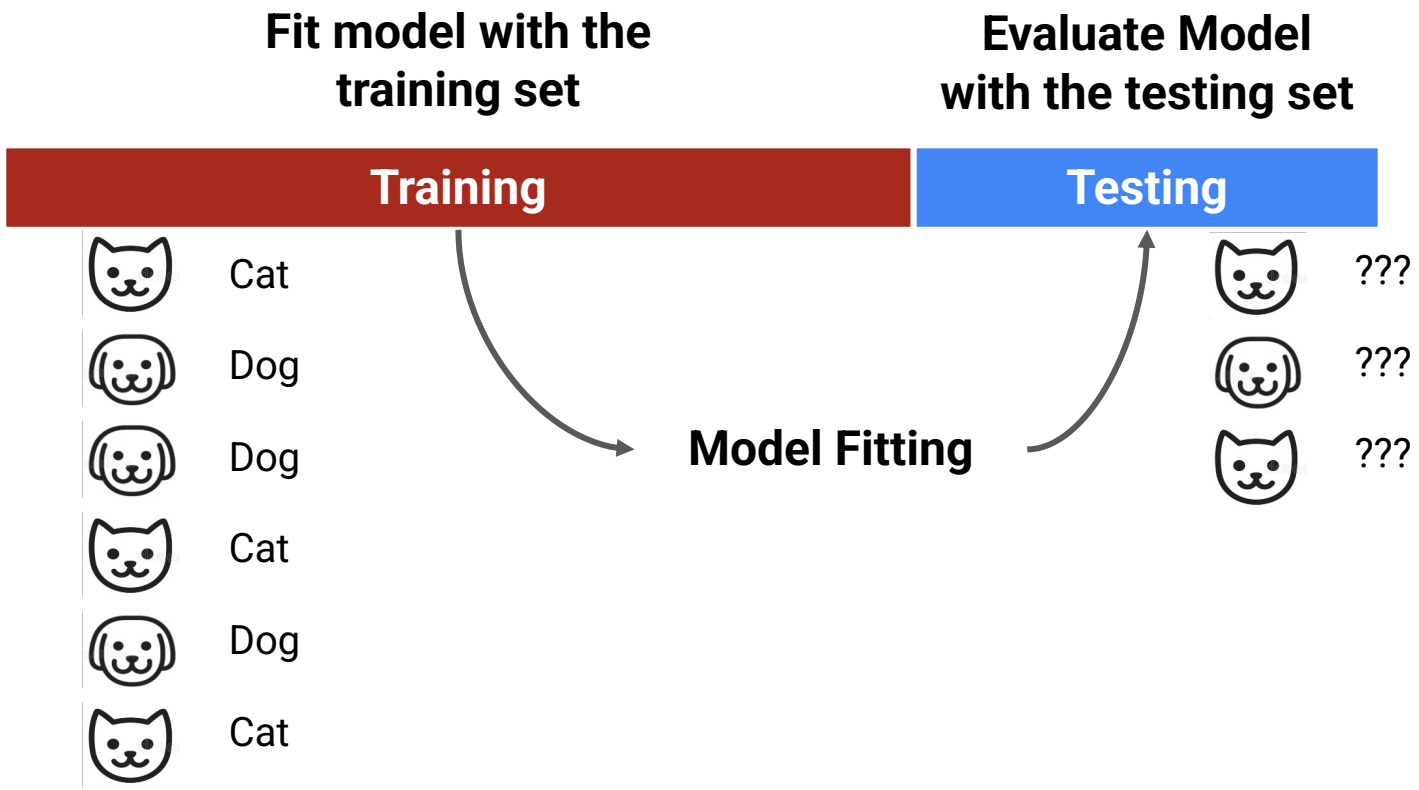
Basic Concepts

Basic Concepts

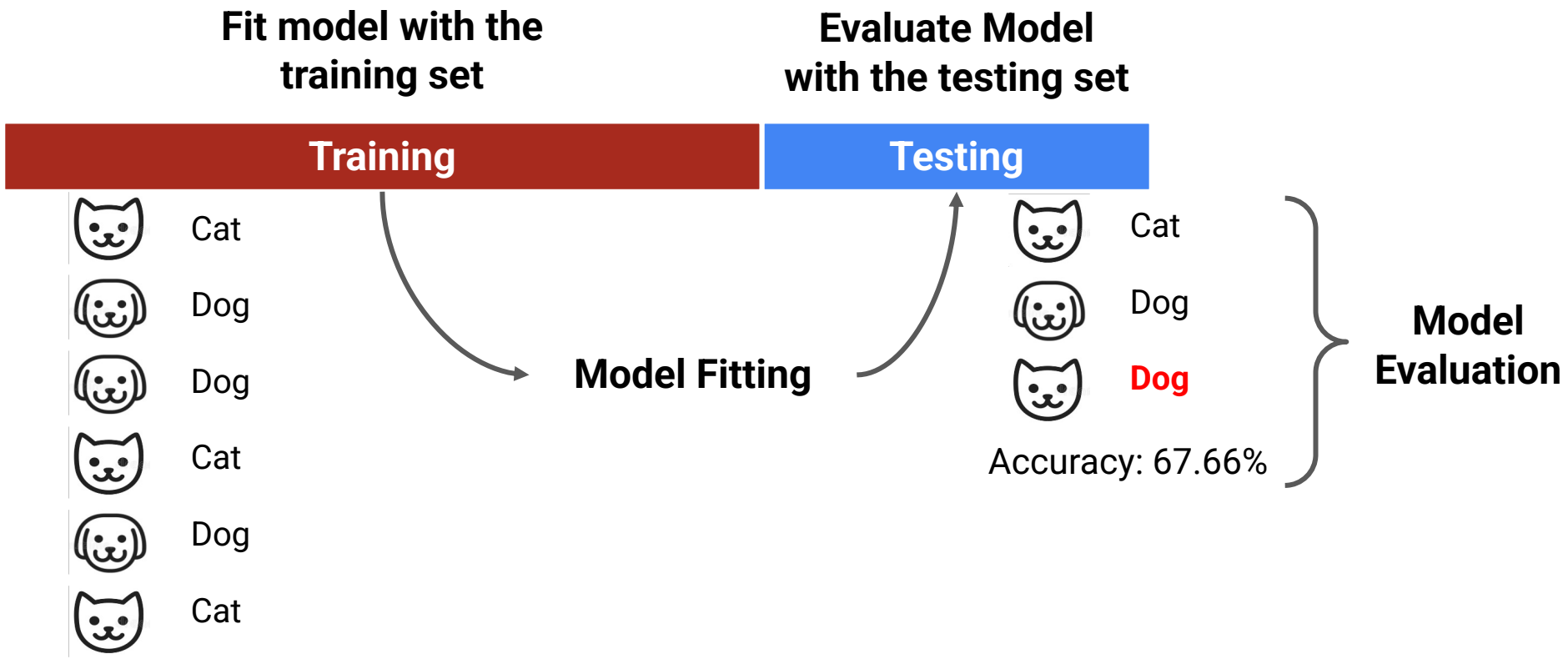


Basic Concepts

Basic Concepts



Basic Concepts



Outline

- Introduction
- Core Concepts
- **Mini Case Study**
- Common Challenges
- Conclusion (The Supervised Learning Process)

Now let’s go to the Google Colab to practice a mini case study.

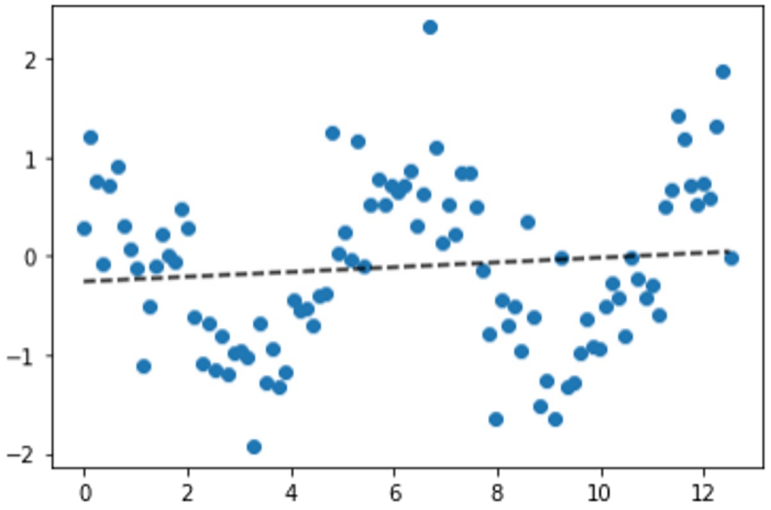
Outline

- Introduction
- Core Concepts
- Mini Case Study
- **Common Challenges**
- Conclusion (The Supervised Learning Process)

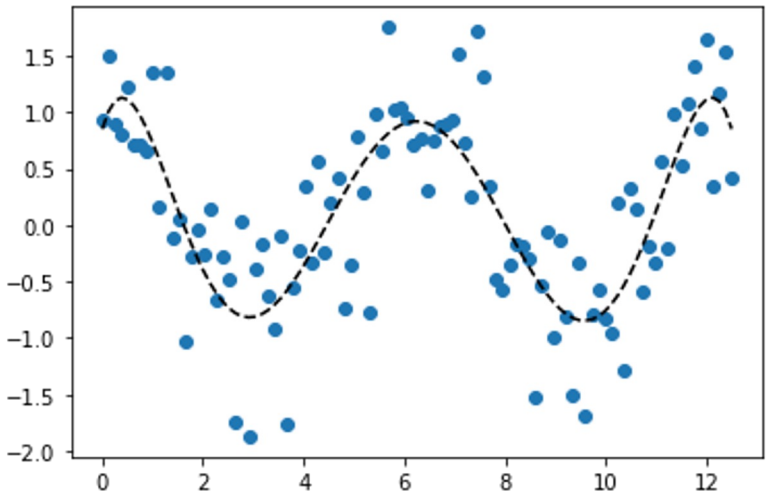
Challenges we faced along the way…

What are the qualities of a “good” model?

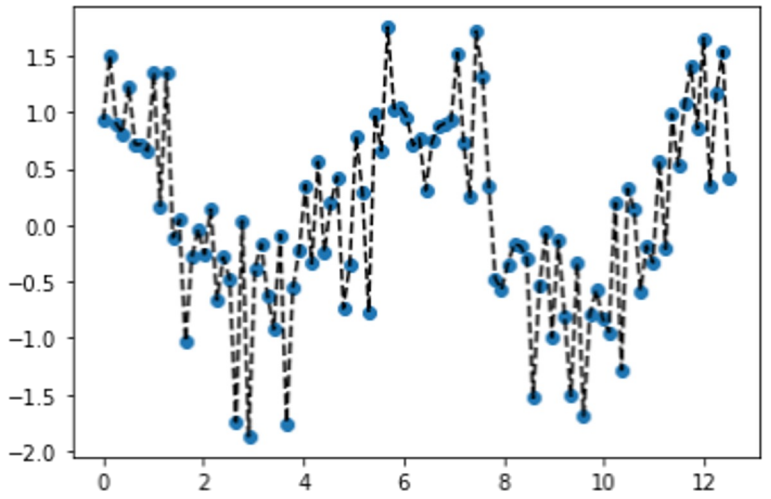
Underfitting and Overfitting



Underfitting



Good Representation of True Model



Overfitting

We want models that can fit the data well without learning anything too specific to your current (training) data

Regularization

Goal

Prevent learning a model that is too “complex” (i.e., overfitting)

Motivation

Model performance can be (1) generalizable to new data and (2) not susceptible to outliers

Approach

Penalize **model coefficients**

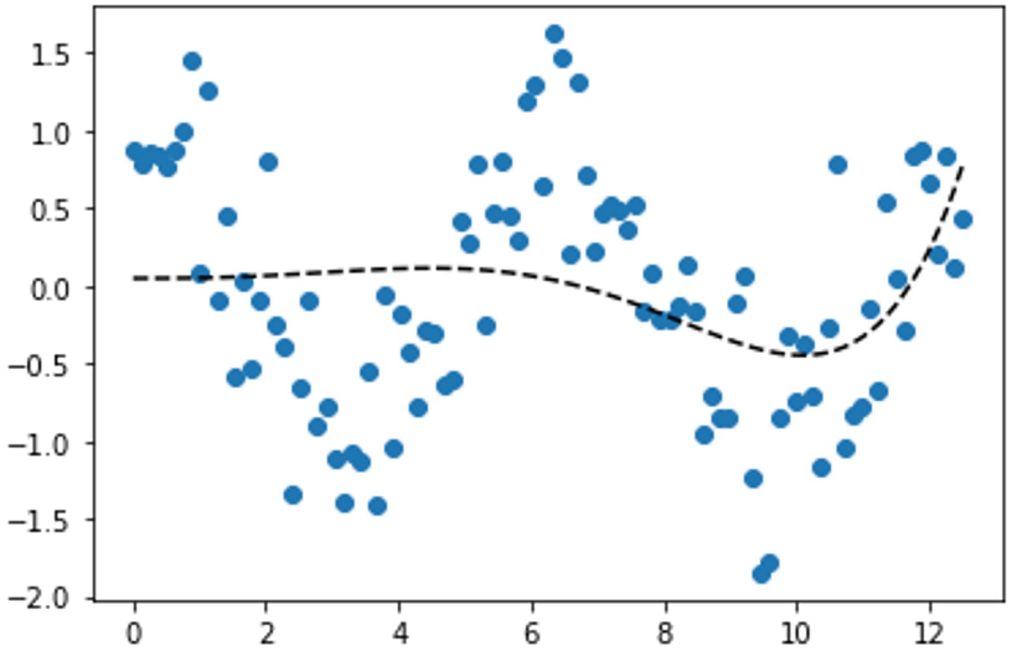
- **Lasso** penalizes the absolute value of coefficients.
- **Ridge** penalizes the squared value of coefficients.

Model coefficients: How much the model performance changes along with every 1-unit change in a specific variable

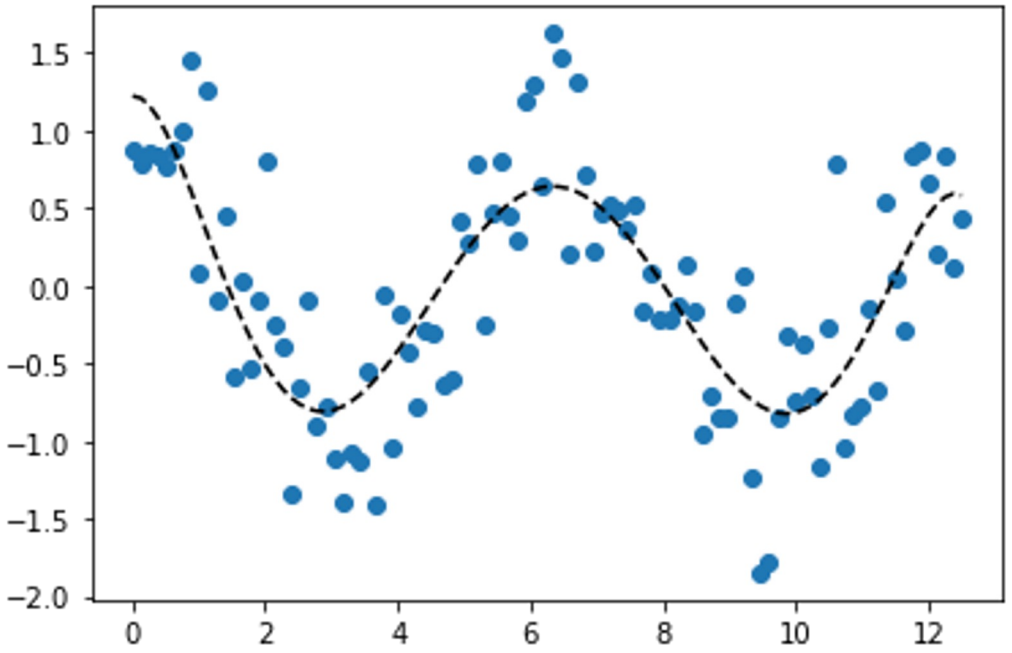
$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_6 x_6$$

Regularization

- Solution for overfitting by penalizing the model coefficients.
- Lasso penalizes the absolute value of coefficients.
- Ridge penalizes the squared value of coefficients.

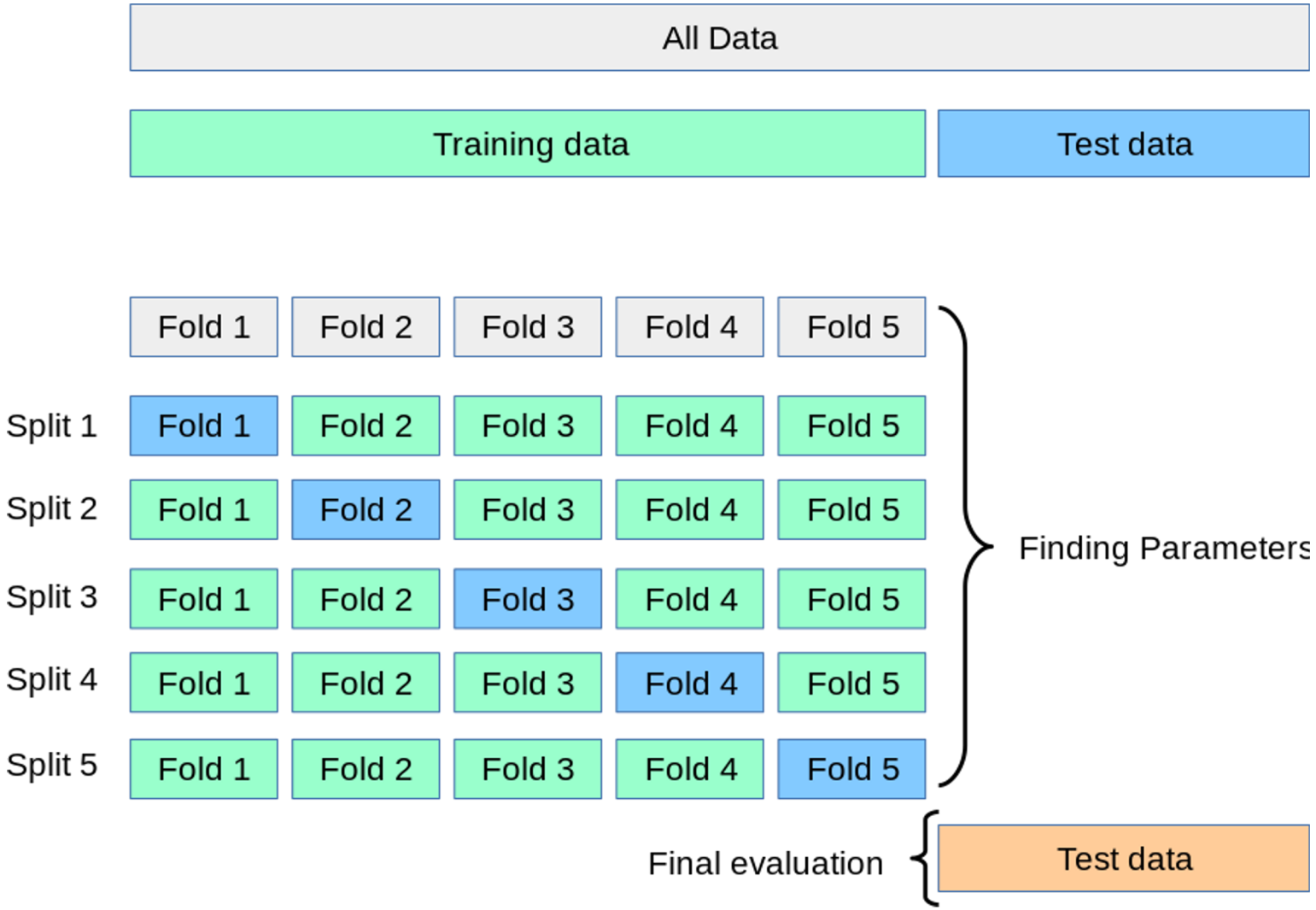


Lasso



Ridge

Cross Validation



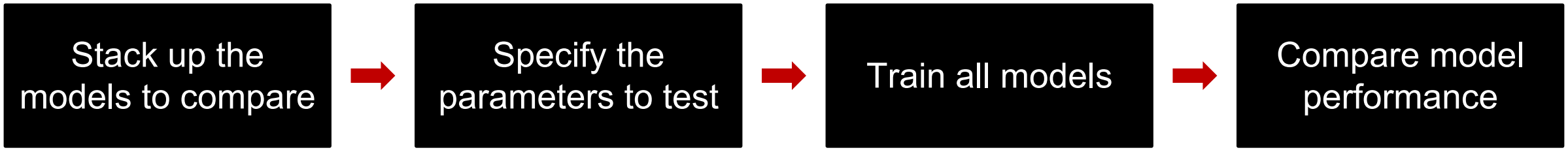
Source: [scikit-learn documentation]

Which model should I choose?

Types of Supervised Learning Models

- Linear Models
  - Linear Regression
  - Logistic Regression
  - Regularized Linear Model (LASSO, Ridge, Elastic Net)
- Tree-based Models
  - Decision Tree
  - Random Forest
  - Gradient Boosting
- Other Models
  - Support Vector Machine
  - Naive Bayes
  - Neural Network

Building Model Pipeline



Building Model Pipeline

Stack up the models to compare → Specify the parameters to test → Train all models in the pipeline → Compare model performance

```
pipelines = {
    'logit': make_pipeline(StandardScaler(),LogisticRegression(penalty = 'none', random_state = 607)),
    'lasso': make_pipeline(StandardScaler(),LogisticRegression(penalty = 'l1', solver = 'liblinear', random_state = 607)),
    'ridge': make_pipeline(StandardScaler(),LogisticRegression(penalty = 'l2', solver = 'liblinear', random_state = 607)),
    'decision-tree': make_pipeline(StandardScaler(),DecisionTreeClassifier(random_state = 607)),
    'random-forest': make_pipeline(StandardScaler(),RandomForestClassifier(random_state = 607)),
}
```

Building Model Pipeline

Stack up the models to compare → **Specify the parameters to test** → Train all models in the pipeline → Compare model performance

```
logit_hyperparameters = {
}

l1_hyperparameters = {
    'logisticregression__C': np.arange(0, 1, 0.1)
}

l2_hyperparameters = {
    'logisticregression__C': np.arange(0, 1, 0.1)
}

dt_hyperparameters = {
    'decisiontreeclassifier__max_depth': [1, 3, 5, 7, 9]
}

rf_hyperparameters = {
    'randomforestclassifier__n_estimators': np.arange(100,500,75)
}
```

Building Model Pipeline

Stack up the models to compare → Specify the parameters to test → **Train all models in the pipeline** → Compare model performance

```
fitted_models = {}

scoring = {'roc_auc': make_scorer(roc_auc_score),
           'accuracy':make_scorer(accuracy_score)}

accuracy = []
roc_auc = []

for key, model in pipelines.items():
    fit_model = GridSearchCV(model, hyperparameters[key], cv=10, n_jobs = -1,
                             scoring = scoring, refit='roc_auc', return_train_score=True)
    fit_model.fit(X_train, y_train)
    fitted_models[key] = fit_model
    cv_accuracy = np.nanmax(fit_model.cv_results_['mean_test_accuracy'])
    cv_roc_auc = np.nanmax(fit_model.cv_results_['mean_test_roc_auc'])
    roc_auc.append(cv_roc_auc)
    accuracy.append(cv_accuracy)
    print(key, 'has been fitted.')
    print('accuracy score:', cv_accuracy)
    print('auc score:', cv_roc_auc)
    print('')
```

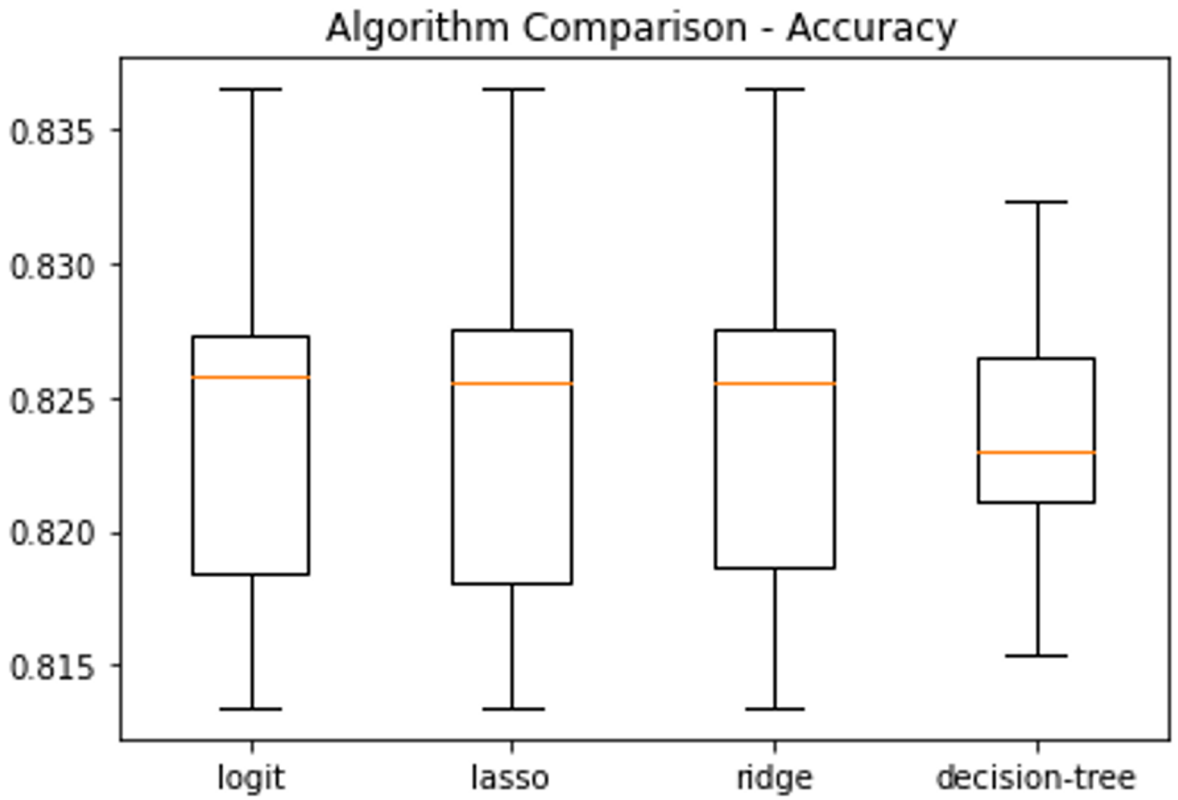
Building Model Pipeline

Stack up the models to compare → Specify the parameters to test → Train all models in the pipeline → **Compare model performance**

```
for key, model in fitted_models.items():
    pred = model.predict(X_test)
    pred_proba = model.predict_proba(X_test)
    pred_proba = [p[1] for p in pred_proba]
    print(key, " :")
    print('AUC Score: ', roc_auc_score(y_test,pred_proba).round(4))
    print('Accuracy: ',  accuracy_score(y_test,pred).round(4))
    print('------------------')
    print('')
```

Understand Model Performance Metrics

- Accuracy
- Confusion Matrix
- ROC Curve



Understand Model Performance Metrics

- Accuracy
- Confusion Matrix
- ROC Curve

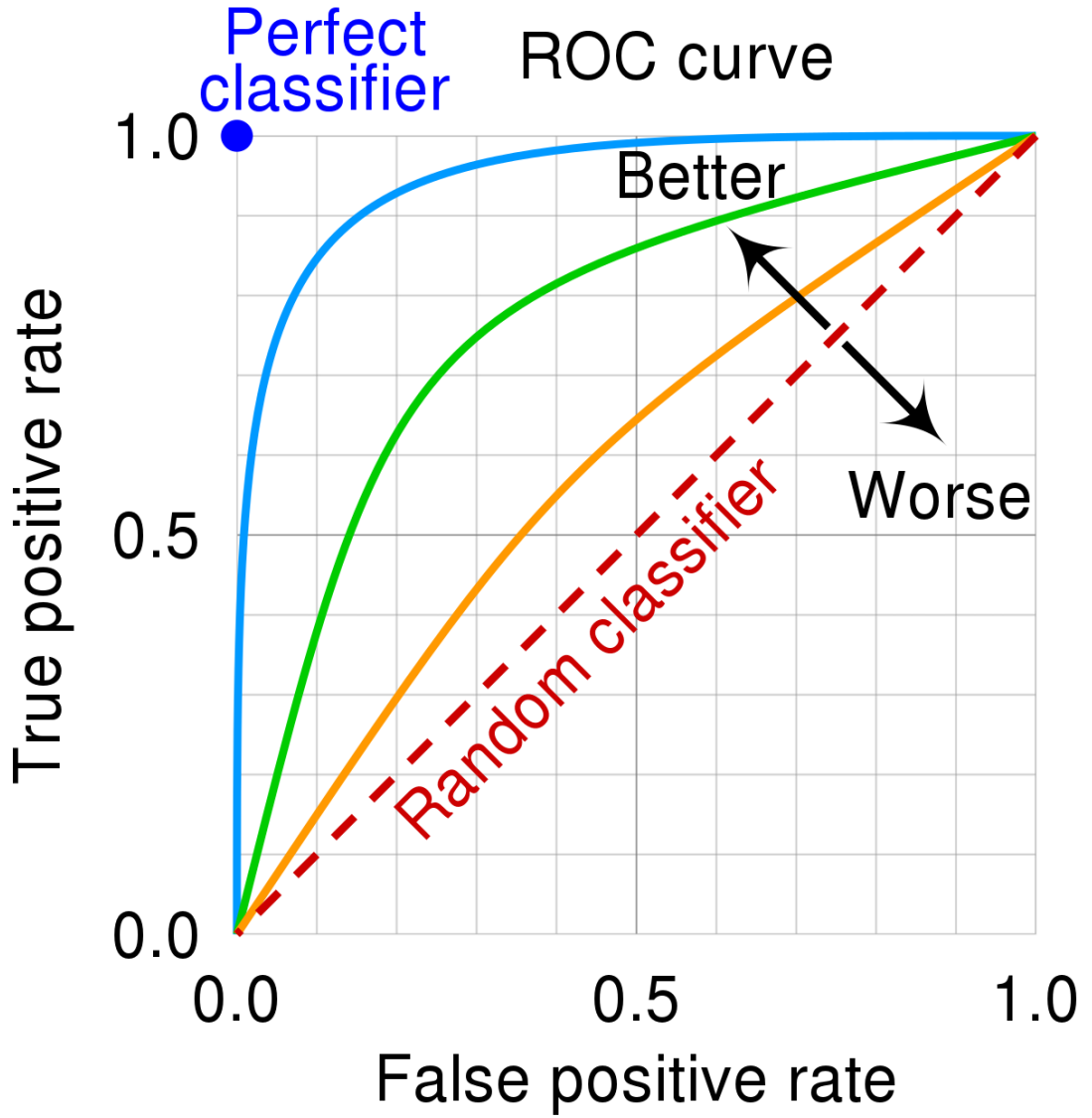
Model Prediction	True Label: 1	True Label: 0
1	True Positive (TP)	False Positive (FP)
0	False Negative (FN)	True Negative (TN)

True Positive Rate = TP/(TP + FN)

False Positive Rate = FP/(FP+ TN)

Understand Model Performance Metrics

- Accuracy
- Confusion Matrix
- ROC Curve



Outline

- Introduction
- Core Concepts
- Mini Case Study
- Common Challenges
- **Conclusion (The Supervised Learning Process)**

Supervised Learning Flowchart

Exploratory Data Analysis → Data Cleaning & Feature Engineering → Algorithm Selection → Model Training → Prediction & Interpretation

Understand the basic information of the dataset.

- Check the distribution of the key variables.
- Learn about the relationship between key variables.

Prepare the clean data for model fitting.

- Deal with missing values, outliers, and data errors.
- Creating dummy variables and/or interaction terms, aggregating data, dropping variables with redundant information, etc.

Select the models for model fitting.

- Think about the practical benefits of selected models.
- Prepare for hyperparameter tuning.

Train the models to select the best model.

- Split data into training and testing set for model evaluation.
- Set up model pipelines.
- Train the model using the training set.
- Select the best model based on model performance.

Make inference from the best model.

- Check the accuracy of the prediction using test set.
- Explain the relationship between model inputs and outputs.
- Integrate models and explanations into theory building process.

Thank you! Questions?

Reach out to schedule a virtual office hour:

Angel Hwang
Senior CCSS Data Science Fellows
hh695@cornell.edu
CCSS-ResearchSupport@cornell.edu