Introducing Create ML

Gaurav Kapoor, Core ML
Lizi Ottens, Core ML
Tao Jia, Core ML
Thank you!
Working with Models

Build your apps with the ready-to-use Core ML models below, or use Core ML Tools to easily convert models into the Core ML format.

Models

MobileNet
MobileNets are based on a streamlined architecture that have depth-wise separable convolutions to build lightweight, deep neural networks.
Detects the dominant objects present in an image from a set of 1000 categories such as trees, animals, food, vehicles, people, and more.
- View original model details
- Download Core ML Model (171 MB)

SqueezeNet
Detects the dominant objects present in an image from a set of 1000 categories such as trees, animals, food, vehicles, people, and more.
With an overall footprint of only 5 MB, SqueezeNet has a similar level of accuracy as AlexNet but with 50 times fewer parameters.
- View original model details
- Download Core ML Model (5 MB)

Model Converters

Core ML Tools
Core ML Tools is a python package that can be used to convert models from machine learning toolboxes into the Core ML format.
- Get Core ML Tools

Apache MXNet
MXNet helps you train machine learning models and convert them into the Core ML format.
- Get MXNet model converter

TensorFlow
Train machine learning models in TensorFlow and easily convert them to the Core ML Model format.
- Get TensorFlow Converter

ONNX
Convert ONNX models you have created to the Core ML Model format.
Continuing our journey...
Create ML
Machine Learning in Swift
Simple model creation
Tailored to your app
Leverages core Apple technologies
Powered by Mac
Data

Images
Data

Images

Text
Data

Images

Text

Tabular Data
Image Classifier

Input

![Input Image]

Output

Label
Tabular Data

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td># &quot; #</td>
<td>Number</td>
</tr>
</tbody>
</table>

**Diagram:**
- Input: # " #
- Output: Number
Work Flow

Problem
Work Flow

Problem

Data

Train
Work Flow

- Problem
- Data
- Train
- Evaluate
Work Flow

Problem → Data → Train → Evaluate → MLMODEL

Diagram showing the workflow of problem, data, training, evaluation, and model.
Work Flow
Work Flow

DataSource
Swift Primitives
DataTable

MLModel
Work Flow

- DataSource
- Swift Primitives
- DataTable
- One line of Code
- Automatic Selection
- Optimized
Work Flow

- DataSource
- Swift Primitives
- DataTable

- One line of Code
- Automatic Selection
- Optimized

- Evaluation Built-in
- Classification Metrics
- Regression Metrics
Work Flow

- DataSource
- Swift Primitives
- DataTable
- One line of Code
- Automatic Selection
- Optimized
- Evaluation Built-in
- Classification Metrics
- Regression Metrics
- write()
Images

Lizi Ottens, Core ML
Image Classification

Strawberry

Orange

Apple

Vision with Core ML

Hall 1

Thursday 3:00PM
Passion Fruit  Blueberry  Raspberry
Data Source

Passion Fruit
- IMG_1764.jpg
- IMG_4765.HEIC
- IMG_4766.png

Blueberry
- IMG_5767.jpg
- IMG_6768.jpg
- IMG_6769.jpg

Raspberry
- IMG_6521.HEIC
- IMG_5622.png
- IMG_5623.HEIC
Transfer Learning

Vision with Core ML

Hall 1

Thursday 3:00PM
Transfer Learning

Labeled Fruits

Vision with Core ML

Hall 1

Thursday 3:00PM
Faster
Faster
Hours
Minutes

10,000 images
Seconds
100 images
Smaller Models
Smaller Models
100s of MB
Demo
Fruit Classifier
Demo Recap
Fruit Classifier
import CreateMLUI

let builder = MLImageClassifierBuilder()
builder.showInLiveView()
import Foundation
import CreateML

// Specify Data
let trainDirectory = URL(fileURLWithPath: "~/Users/createml/Desktop/Fruits")
let testDirectory = URL(fileURLWithPath: "~/Users/createml/Desktop/TestFruits")

// Create Model
let classifier = try MLImageClassifier(trainingData: .labeledDirectories(at: trainDirectory))

// Evaluate Model
let evaluation = classifier.evaluation(on: .labeledDirectories(at: testDirectory))
import Foundation
import CreateML

// Specify Data
let trainDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/Fruits")
let testDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/TestFruits")

// Create Model
let model = try MLImageClassifier(trainingData: .labeledDirectories(at: trainDirectory))

// Evaluate Model
let evaluation = model.evaluation(on: .labeledDirectories(at: testDirectory))

// Save Model
try model.write(to: URL(fileURLWithPath: "/Users/createml/Desktop/FruitClassifier.mlmodel"))
import Foundation
import CreateML

// Specify Data
let trainDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/Fruits")
let testDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/TestFruits")

// Create Model
let model = try MLImageClassifier(trainingData: .labeledDirectories(at: trainDirectory))

// Evaluate Model
let evaluation = model.evaluation(on: .labeledDirectories(at: testDirectory))

// Save Model
try model.write(to: URL(fileURLWithPath: "/Users/createml/Desktop/FruitClassifier.mlmodel"))
import Foundation
import CreateML

// Specify Data
let trainDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/Fruits")
let testDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/TestFruits")

// Create Model
let model = try MLImageClassifier(trainingData: .labeledDirectories(at: trainDirectory))

// Evaluate Model
let evaluation = model.evaluation(on: .labeledDirectories(at: testDirectory))

// Save Model
try model.write(to: URL(fileURLWithPath: "/Users/createml/Desktop/FruitClassifier.mlmodel"))
import Foundation
import CreateML

// Specify Data
let trainDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/Fruits")
let testDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/TestFruits")

// Create Model
let model = try MLImageClassifier(trainingData: .labeledDirectories(at: trainDirectory))

// Evaluate Model
let evaluation = model.evaluation(on: .labeledDirectories(at: testDirectory))

// Save Model
try model.write(to: URL(fileURLWithPath: "/Users/createml/Desktop/FruitClassifier.mlmodel"))
import Foundation
import CreateML

// Specify Data
let trainDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/Fruits")
let testDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/TestFruits")

// Create Model
let model = try MLImageClassifier(trainingData: .labeledDirectories(at: trainDirectory))

// Evaluate Model
let evaluation = model.evaluation(on: .labeledDirectories(at: testDirectory))

// Save Model
try model.write(to: URL(fileURLWithPath: "/Users/createml/Desktop/FruitClassifier.mlmodel"))
import Foundation
import CreateML

// Specify Data
let trainDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/Fruits")
let testDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/TestFruits")

// Create Model
let model = try MLImageClassifier(trainingData: .labeledDirectories(at: trainDirectory))

// Evaluate Model
let evaluation = model.evaluation(on: .labeledDirectories(at: testDirectory))

// Save Model
try model.write(to: URL(fileURLWithPath: "/Users/createml/Desktop/FruitClassifier.mlmodel"))
import Foundation
import CreateML

// Specify Data
let trainDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/Fruits"
let testDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/TestFruits"

// Create Model
let model = try MLImageClassifier(trainingData: .labeledDirectories(at: trainDirectory)

// Evaluate Model
let evaluation = model.evaluation(on: .labeledDirectories(at: testDirectory)

// Save Model
try model.write(to: URL(fileURLWithPath: "/Users/createml/Desktop/FruitClassifier.mlmodel"))
$ swift fruitClassifier.swift

$ chmod u+x ./fruitClassifier.swift

$ ./fruitClassifier.swift
$ swift
Welcome to Apple Swift version 4.2.
  1 > import CreateML
  2 >
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Introducing Natural Language Framework

Hall 3

Wednesday 4:00PM
The energy of developers is amazing!
Deposed Prince wants to give you money

SPAM

Spam Analysis
The Warriors just had an amazing comeback win

SPORT

Topic Analysis
Data
I'm looking for a place to stay in Barcelona, HOTELS
Where can I get good Mexican food on a Sunday?, RESTAURANTS
Find me an inexpensive round trip flight to London., FLIGHTS
...
Text Classification
(Typical work flow)

Introducing Natural Language Framework

Label

Hall 3

Wednesday 4:00PM
Text Classification
(Typical work flow)

Introducing Natural Language Framework

Hall 3
Wednesday 4:00PM
Text Classification
(Typical work flow)

Introducing Natural Language Framework

Hall 3
Wednesday 4:00PM
Text Classification
(Typical work flow)

Introducing Natural Language Framework
Hall 3
Wednesday 4:00PM
Text Classification
(Typical work flow)
Text Classification
(Typical work flow)

Introducing Natural Language Framework
Hall 3
Wednesday 4:00PM
Text Classification with Create ML
Text Classification with Create ML

Introducing Natural Language Framework

Hall 3

Wednesday 4:00PM
Text Classification with Create ML

Introducing Natural Language Framework

Hall 3

Wednesday 4:00PM
Example
Stay Positive

Encouraging positive posts
Stay Positive
Encouraging positive posts

I hate traffic

Post
Clear
Stay Positive
Encouraging positive posts

I love driving my car at 5mph...just chilling in traffic.
Demo

Text Classifier
Demo Recap

Text Classifier
import CreateML
import Foundation

// Specify Data
let trainDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/train")
let testDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/test")

// Create Model
let classifier = try MLTextClassifier(trainingData: .labeledDirectories(at: trainDirectory))

// Evaluate Model
let evaluation = classifier.evaluation(on: .labeledDirectories(at: testDirectory))

// Save Model
let modelPath = URL(fileURLWithPath: "/Users/createml/Desktop/TextClassifier.mlmodel")
try classifier.write(to: modelPath)
import CreateML
import Foundation

// Specify Data
let trainDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/train")
let testDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/test")

// Create Model
let classifier = try MLTextClassifier(trainingData: .labeledDirectories(at: trainDirectory))

// Evaluate Model
let evaluation = classifier.evaluation(on: .labeledDirectories(at: testDirectory))

// Save Model
let modelPath = URL(fileURLWithPath: "/Users/createml/Desktop/TextClassifier.mlmodel")
try classifier.write(to: modelPath)
import CreateML
import Foundation

// Specify Data
let trainDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/train")
let testDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/test")

// Create Model
let classifier = try MLTextClassifier(trainingData: .labeledDirectories(at: trainDirectory))

// Evaluate Model
let evaluation = classifier.evaluation(on: .labeledDirectories(at: testDirectory))

// Save Model
let modelPath = URL(fileURLWithPath: "/Users/createml/Desktop/TextClassifier.mlmodel")
try classifier.write(to: modelPath)
import CreateML
import Foundation

// Specify Data
let trainDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/train")
let testDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/test")

// Create Model
let classifier = try MLTextClassifier(trainingData: .labeledDirectories(at: trainDirectory))

// Evaluate Model
let evaluation = classifier.evaluation(on: .labeledDirectories(at: testDirectory))

// Save Model
let modelPath = URL(fileURLWithPath: "/Users/createml/Desktop/TextClassifier.mlmodel")
try classifier.write(to: modelPath)
import CreateML
import Foundation

// Specify Data
let trainDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/train")
let testDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/test")

// Create Model
let classifier = try MLTextClassifier(trainingData: .labeledDirectories(at: trainDirectory))

// Evaluate Model
let evaluation = classifier.evaluation(on: .labeledDirectories(at: testDirectory))

// Save Model
let modelPath = URL(fileURLWithPath: "/Users/createml/Desktop/TextClassifier.mlmodel")
try classifier.write(to: modelPath)
import CreateML
import Foundation

// Specify Data
let trainDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/train")
let testDirectory = URL(fileURLWithPath: "/Users/createml/Desktop/test")

// Create Model
let classifier = try MLTextClassifier(trainingData: .labeledDirectories(at: trainDirectory))

// Evaluate Model
let evaluation = classifier.evaluation(on: .labeledDirectories(at: testDirectory))

// Save Model
let modelPath = URL(fileURLWithPath: "/Users/createml/Desktop/TextClassifier.mlmodel")
try classifier.write(to: modelPath)
Tabular Data
Predict house price using:

- No. of bedrooms
- No. of baths
- Location
- Sq. ft
- Lot size
Wine Quality using:
- Acidity
- Sugar
- pH
- Alcohol
- Citric
<table>
<thead>
<tr>
<th>Rating</th>
<th>Price</th>
<th>Happy hour</th>
<th>Distance</th>
</tr>
</thead>
</table>

Where to go tonight
Data
<table>
<thead>
<tr>
<th>beds</th>
<th>baths</th>
<th>squareFt</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>2000</td>
<td>400K</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2500</td>
<td>500K</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1800</td>
<td>450K</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1500</td>
<td>300K</td>
</tr>
</tbody>
</table>
## Tabular Data

### Example

<table>
<thead>
<tr>
<th>beds</th>
<th>baths</th>
<th>squareFt</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>2000</td>
<td>400K</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2500</td>
<td>500K</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1800</td>
<td>450K</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1500</td>
<td>300K</td>
</tr>
<tr>
<td>Feature</td>
<td>beds</td>
<td>baths</td>
<td>squareFt</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>2000</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3</td>
<td>2500</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>1800</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>1500</td>
</tr>
</tbody>
</table>
## Tabular Data

<table>
<thead>
<tr>
<th>beds</th>
<th>baths</th>
<th>squareFt</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>2000</td>
<td>400K</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2500</td>
<td>500K</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1800</td>
<td>450K</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1500</td>
<td>300K</td>
</tr>
<tr>
<td>beds</td>
<td>baths</td>
<td>squareFt</td>
<td>Price</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2000</td>
<td>400K</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2500</td>
<td>500K</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1800</td>
<td>450K</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1500</td>
<td>300K</td>
</tr>
</tbody>
</table>
Common Sources

**Files**

- **CSV**

- **JSON**

```plaintext
beds,baths,squareFt,price
2,2,2000,400000
4,3,2500,500000
3,2,1800,450000
3,2,1500,300000
```

**Code**

```swift
func loadData() -> [String: [Any]] {
    let data = [
        
            
        ]

    return data
}
```
<table>
<thead>
<tr>
<th>bedrooms</th>
<th>bathrooms</th>
<th>sqFt</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>2000</td>
<td>400K</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2500</td>
<td>500K</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1800</td>
<td>450K</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1500</td>
<td>300K</td>
</tr>
</tbody>
</table>
// You can read data
let houseData = try MLDataTable(contentsOf: mycsv)
// You can read data
let houseData = try MLDataTable(contentsOf: mycsv)

// You can access the Column
houseData["price"]
// You can add, subtract, multiply, divide two columns

<table>
<thead>
<tr>
<th>sqFt</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>400K</td>
</tr>
<tr>
<td>2500</td>
<td>500K</td>
</tr>
<tr>
<td>1800</td>
<td>450K</td>
</tr>
<tr>
<td>1500</td>
<td>300K</td>
</tr>
</tbody>
</table>
// You can add, subtract, multiply, divide two columns

```javascript
let pricePerSqft = houseData["price"] / houseData["sqft"]
```

<table>
<thead>
<tr>
<th>sqFt</th>
<th>Price</th>
<th>pricePerSqFt</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>400K</td>
<td>200</td>
</tr>
<tr>
<td>2500</td>
<td>500K</td>
<td>200</td>
</tr>
<tr>
<td>1800</td>
<td>450K</td>
<td>250</td>
</tr>
<tr>
<td>1500</td>
<td>300K</td>
<td>200</td>
</tr>
</tbody>
</table>
// You can filter on a column

<table>
<thead>
<tr>
<th>bedrooms</th>
<th>bathrooms</th>
<th>sqFt</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>2000</td>
<td>400K</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2500</td>
<td>500K</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1800</td>
<td>450K</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1500</td>
<td>300K</td>
</tr>
</tbody>
</table>
// You can filter on a column
let indexOfBigHouses = houseData["sqft"] >= 2000
let largeHouseData = houseData[indexOfBigHouses]
// You can make a model

let classifier = try MLLinearRegressor(trainingData: houseData, targetColumn: "price")
// You can make a model

let classifier = try MLBoostedTreeRegressor(trainingData: houseData, targetColumn: "price")
// You can make a model

define classifier = try MLRandomForestRegressor(trainingData: houseData, targetColumn: "price")
// You can make a model

let classifier = try (trainingData: houseData, targetColumn: "price")
// You can make a model

let classifier = try MLRegressor(trainingData: houseData, targetColumn: "price")
import CreateML

// Specify Data
let trainingCSV = URL(fileURLWithPath: "~/Users/createml/HouseData.csv")
let houseData = MLDataTable(contentsOf: trainingCSV)
let (trainingData, testData) = houseData.randomSplit(by: 0.8, seed: 0)

// Create Model
let pricer = try MLRegressor(trainingData: houseData, targetColumn: "price")

// Evaluate Model
let metrics = try pricer.testingMetrics(on: testData)

// Save Model
try pricer.write(to: URL(fileURLWithPath: "~/Users/createml/HousePricer.mlmodel"))
```swift
import CreateML

// Specify Data
let trainingCSV = URL(fileURLWithPath: "/Users/createml/HouseData.csv")
let houseData = MLDDataTable(contentsOf: trainingCSV)
let (trainingData,testData) = houseData.randomSplit(by: 0.8, seed: 0)

// Create Model
let pricer = try MLRegressor(trainingData: houseData, targetColumn: "price")

// Evaluate Model
let metrics = try pricer.testingMetrics(on: testData)

// Save Model
try pricer.write(to: URL(fileURLWithPath: "/Users/createml/HousePricer.mlmodel"))
```
import CreateML

// Specify Data
let trainingCSV = URL(fileURLWithPath: "~/Users/createml/HouseData.csv")
let houseData = MLDataTable(contentsOf: trainingCSV)
let (trainingData,testData) = houseData.randomSplit(by: 0.8, seed: 0)

// Create Model
let pricer = try MLRegressor(trainingData: houseData, targetColumn: "price")

// Evaluate Model
let metrics = try pricer.testingMetrics(on: testData)

// Save Model
try pricer.write(to: URL(fileURLWithPath: "~/Users/createml/HousePricer.mlmodel"))
import CreateML

// Specify Data
let trainingCSV = URL(fileURLWithPath: "~/Users/createml/HouseData.csv")
let houseData = MLDataTable(contentsOf: trainingCSV)
let (trainingData, testData) = houseData.randomSplit(by: 0.8, seed: 0)

// Create Model
let pricer = try MLRegressor(trainingData: houseData, targetColumn: "price")

// Evaluate Model
let metrics = try pricer.testingMetrics(on: testData)

// Save Model
try pricer.write(to: URL(fileURLWithPath: "~/Users/createml/HousePricer.mlmodel"))
import CreateML

// Specify Data
let trainingCSV = URL(fileURLWithPath: "~/Users/createml/HouseData.csv")
let houseData = MLDataTable(contentsOf: trainingCSV)
let (trainingData, testData) = houseData.randomSplit(by: 0.8, seed: 0)

// Create Model
let pricer = try MLRegressor(trainingData: houseData, targetColumn: "price")

// Evaluate Model
let metrics = try pricer.testingMetrics(on: testData)

// Save Model
try pricer.write(to: URL(fileURLWithPath: "~/Users/createml/HousePricer.mlmodel"))
import CreateML

// Specify Data
let trainingCSV = URL(fileURLWithPath: "~/Users/createml/HouseData.csv")
let houseData = MLDataTable(contentsOf: trainingCSV)
let (trainingData,testData) = houseData.randomSplit(by: 0.8, seed: 0)

// Create Model
let pricer = try MLRegressor(trainingData: houseData, targetColumn: "price")

// Evaluate Model
let metrics = try pricer.testingMetrics(on: testData)

// Save Model
try pricer.write(to: URL(fileURLWithPath: "~/Users/createml/HousePricer.mlmodel"))
ML framework in Swift
Simple and Powerful
Leverages core Apple technologies
Work Flow

Problem → Data → Train → Evaluate → MLOps
Available on macOS Mojave
Try it out
Machine Learning Get Together
Labs
Related Sessions in the WWDC App
## More Information

https://developer.apple.com/wwdc18/703

<table>
<thead>
<tr>
<th>Event</th>
<th>Room</th>
<th>Date</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine Learning Lab</td>
<td>Technology Lab 2</td>
<td>Wednesday</td>
<td>4:00PM</td>
</tr>
<tr>
<td>Machine Learning Lab</td>
<td>Technology Lab 12</td>
<td>Friday</td>
<td>2:00PM</td>
</tr>
</tbody>
</table>