

# CS 615 – Deep Learning

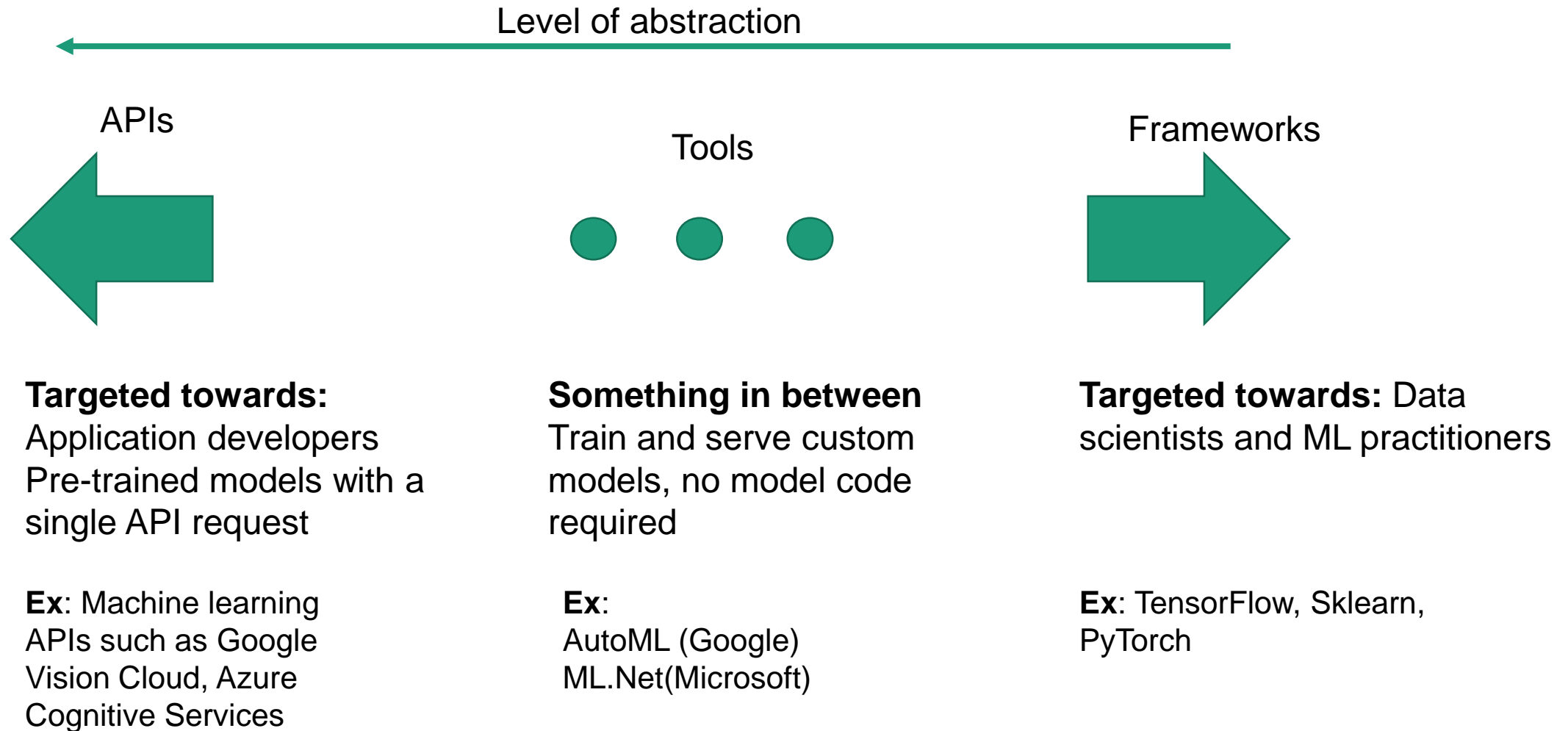
ML/DL APIs

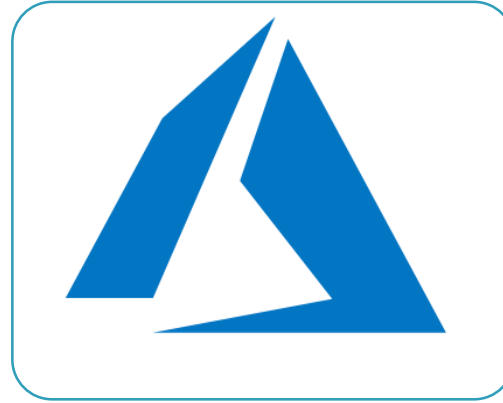
Content largely created by Maryam Daniali

# ML/DL APIs

- From the onset of this class we said we'd be learning the foundations of deep learning.
  - And therefore learning where they came from and implementing some from scratch.
  - After all, as computer scientists we should be the ones coming up with these algorithms!
- Of course, often “off the shelf” existing solutions may be useful
  - Or at least a starting point.
- So let's start by doing a quick survey of some of the most popular ML/DL APIs.

# Machine learning tools





## 3 Most Well-known ML APIs

### Google Cloud Machine Learning API

+Good amount of data for pre-training

You can code your own model.

Also, contains 5 APIs that give access to pre-trained models to accomplish common machine learning tasks

- 1) Cloud Vision
- 2) Cloud Video Intelligence
- 3) Cloud Speech
- 4) Cloud Natural Language
- 5) Cloud Translation

### Microsoft Azure Machine Learning API

+Easy for beginners to work with

You can code your own model or use services for ML with less code needed.

### Amazon Machine Learning API

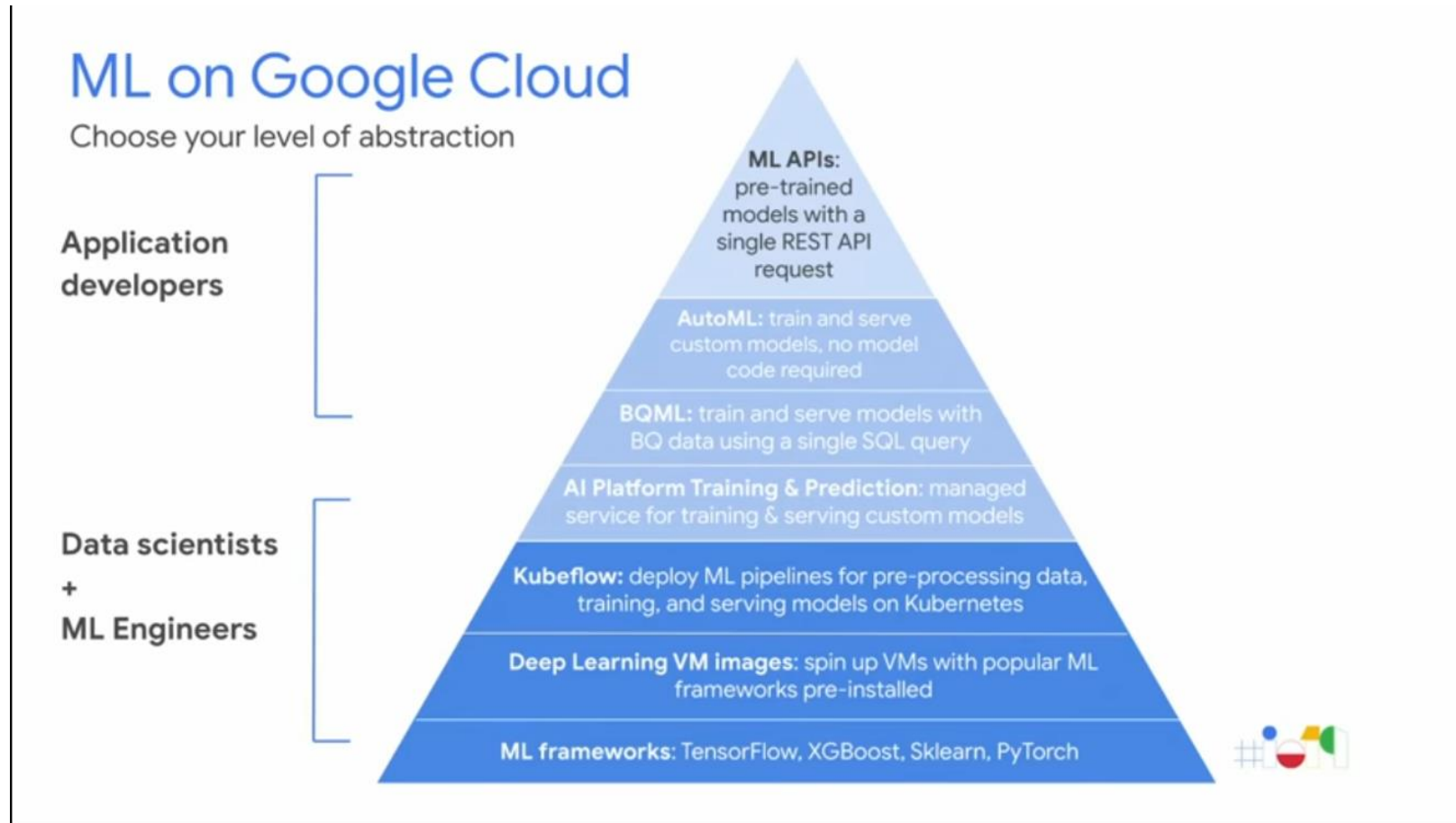
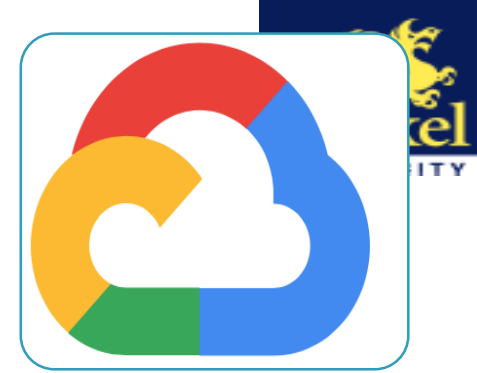
+Affordable prices, many functions for free

Pay as you go (only for what you use).

You can code your own model or use services for ML with less code needed.

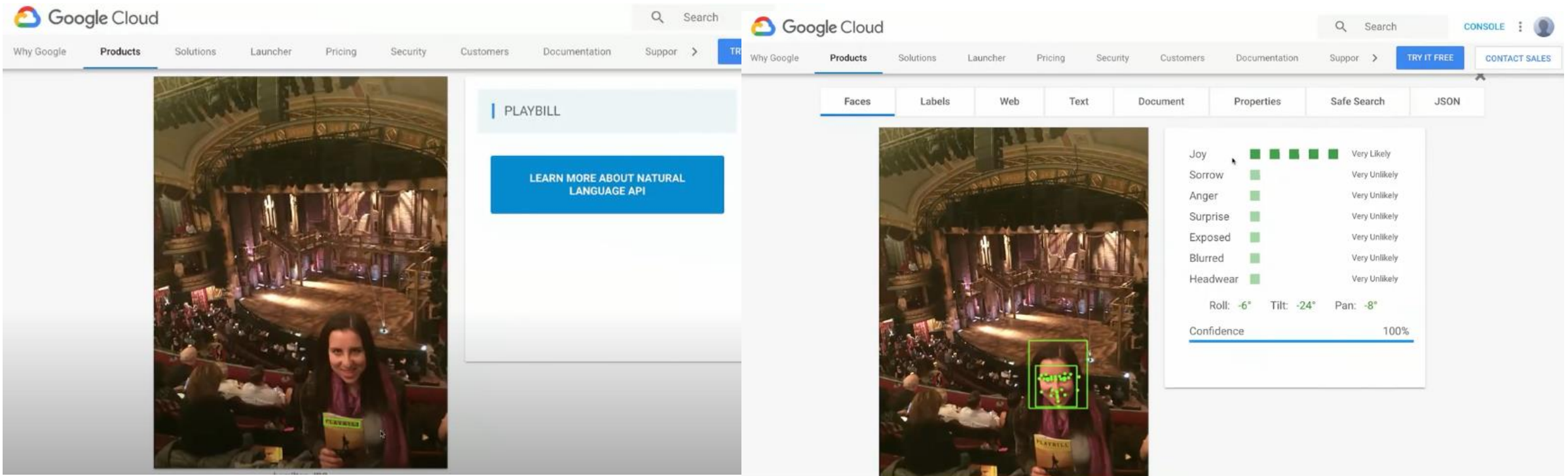
<https://rapidapi.com/blog/top-machine-learning-apis/>

# Google Cloud ML abstraction Hierarchy



# Results of Google Cloud Vision API on a sample image

- Based on the feature detection in the request, it detected the face and the feeling as well as the text on the playbill.



The screenshot displays the Google Cloud Vision API interface. On the left, a sample image of a theater interior with a woman in the foreground holding a playbill is shown. On the right, the analysis results are displayed under the 'Faces' tab. The results include a list of emotions and their likelihoods, as well as camera parameters and a confidence score.

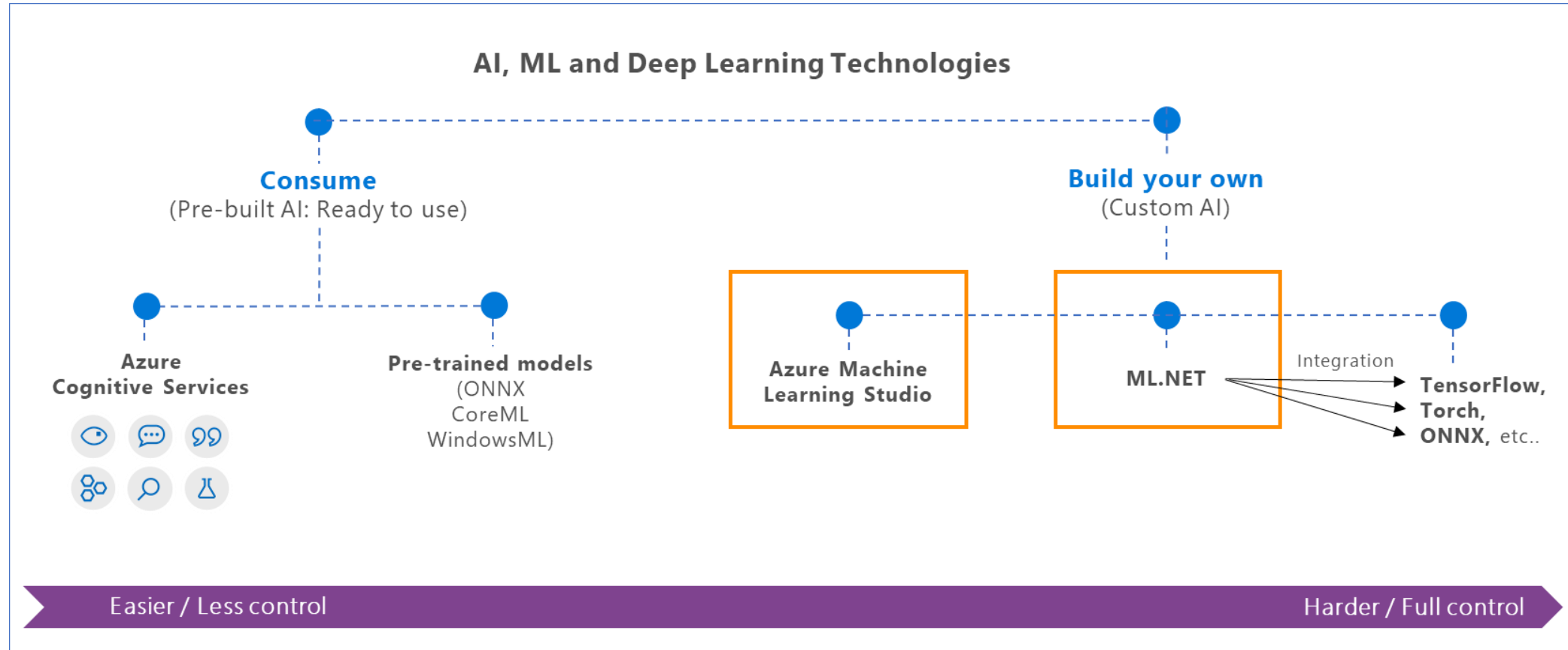
Feature	Confidence	Label
Joy	Very High	Very Likely
Sorrow	Low	Very Unlikely
Anger	Low	Very Unlikely
Surprise	Low	Very Unlikely
Exposed	Low	Very Unlikely
Blurred	Low	Very Unlikely
Headwear	Low	Very Unlikely

Roll: -6° Tilt: -24° Pan: -8°

Confidence 100%



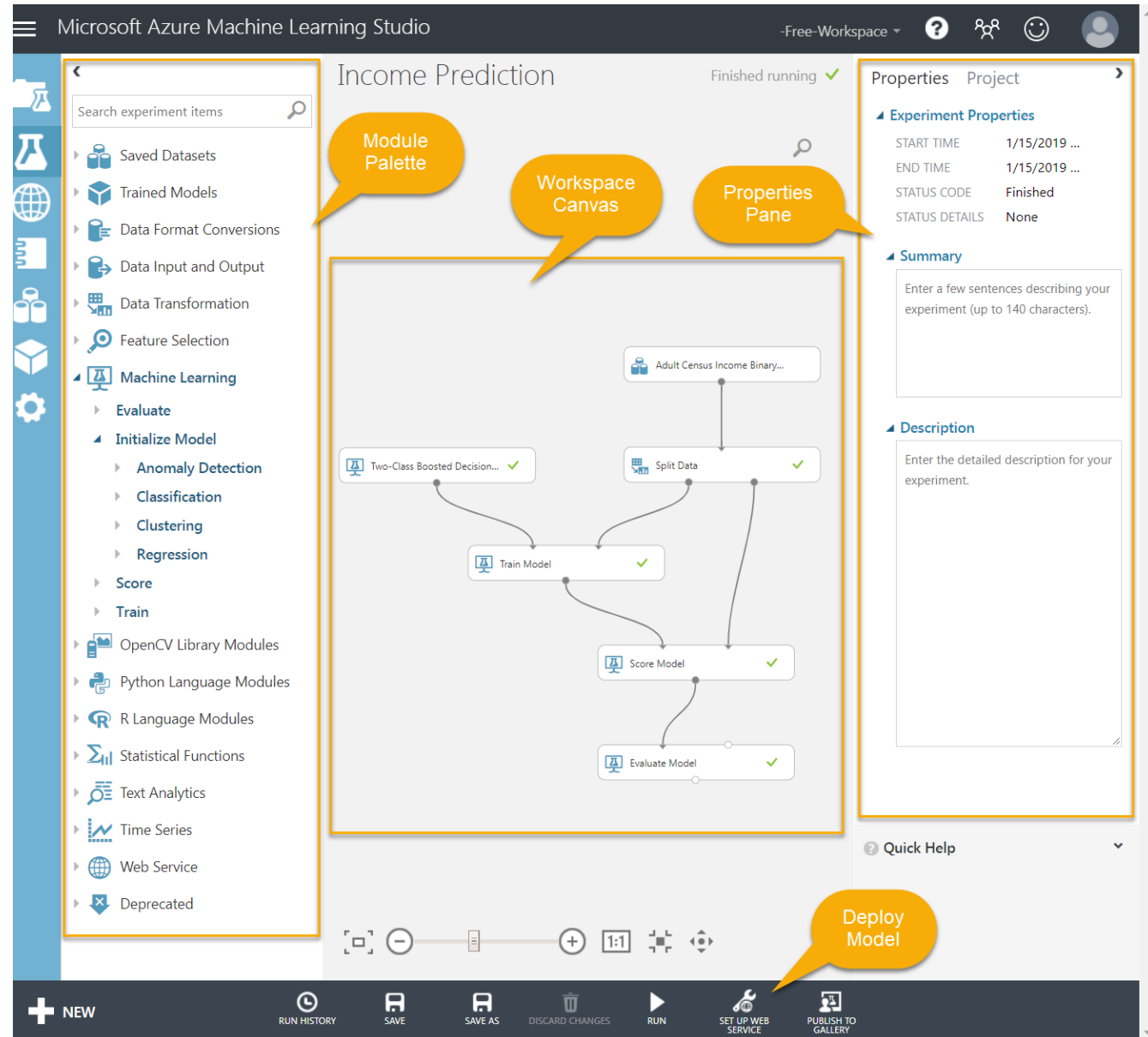
# Microsoft Azure ML Hierarchy



<https://devblogs.microsoft.com/premier-developer/custom-ai-models-with-azure-machine-learning-studio-and-ml-net/>

# Azure Machine Learning Studio

- A drag-and-drop graphical user interface



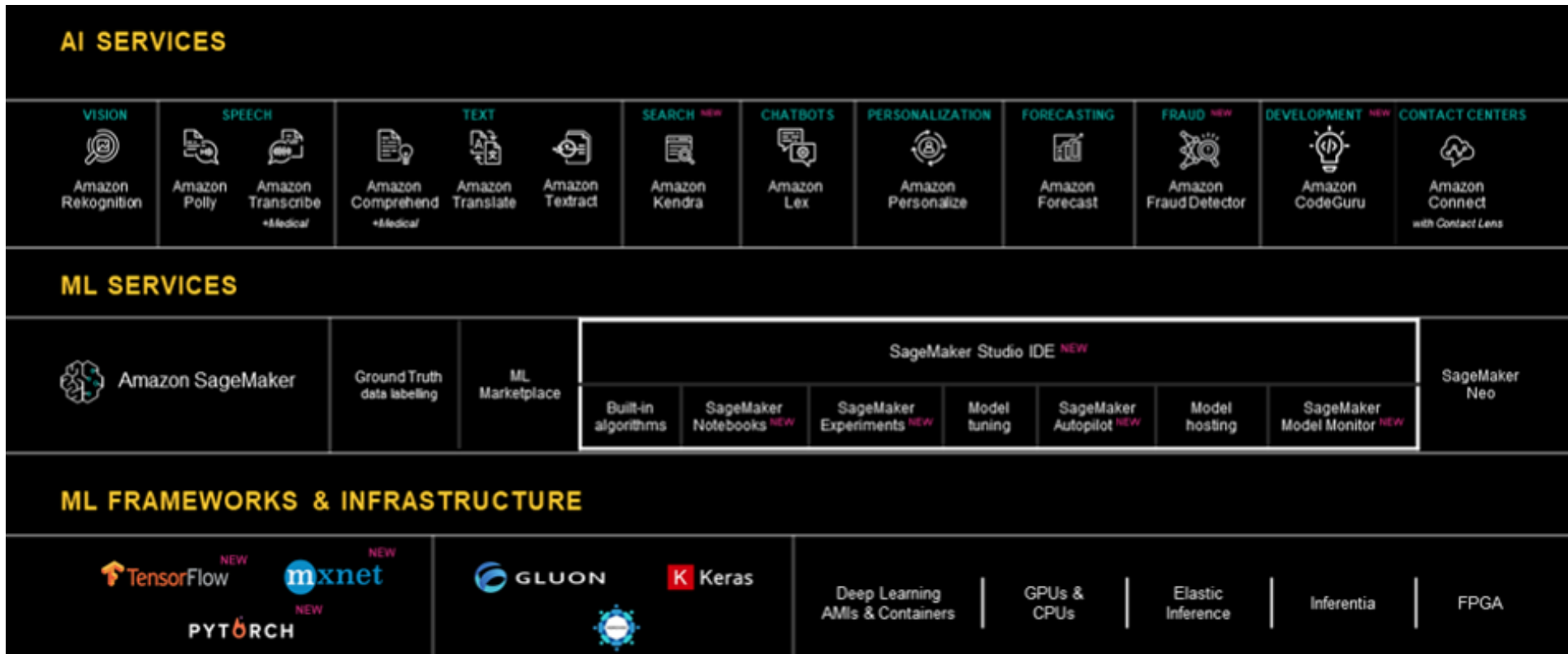
The screenshot displays the Microsoft Azure Machine Learning Studio interface for an experiment titled "Income Prediction". The interface is divided into several key sections:

- Module Palette (Left):** A sidebar containing a search bar and a list of experiment items categorized by type: Saved Datasets, Trained Models, Data Format Conversions, Data Input and Output, Data Transformation, Feature Selection, Machine Learning (with sub-categories like Evaluate, Initialize Model, Anomaly Detection, Classification, Clustering, Regression, Score, and Train), OpenCV Library Modules, Python Language Modules, R Language Modules, Statistical Functions, Text Analytics, Time Series, Web Service, and Deprecated.
- Workspace Canvas (Center):** The main area where the experiment workflow is built. It shows a sequence of modules: "Adult Census Income Binary..." (dataset), "Split Data" (data manipulation), "Two-Class Boosted Decision..." (model), "Train Model" (training), "Score Model" (scoring), and "Evaluate Model" (evaluation). Each module has a green checkmark indicating it has been successfully executed.
- Properties Pane (Right):** A panel showing details for the selected experiment. It includes "Experiment Properties" (Start Time: 1/15/2019, End Time: 1/15/2019, Status Code: Finished, Status Details: None) and a "Summary" section with a text area for describing the experiment (up to 140 characters).
- Bottom Bar:** A toolbar with icons for "NEW", "RUN HISTORY", "SAVE", "SAVE AS", "DISCARD CHANGES", "RUN", "SET UP WEB SERVICE", and "PUBLISH TO GALLERY".

Orange callout boxes highlight specific features: "Module Palette" points to the left sidebar, "Workspace Canvas" points to the central workflow area, "Properties Pane" points to the right sidebar, and "Deploy Model" points to the "SET UP WEB SERVICE" button in the bottom bar.



# AWS ML Hierarchy



<https://specials-images.forbesimg.com/imageserve/5e6017d8e1e617000758dc82/960x0.jpg?fit=scale>

# Ex: Amazon Comprehend (NLP)

## API explorer

Paste the text that you would like to analyze with natural language processing.

[Clear text](#)

A suspect bit the ear of a 4-year-old police dog and injured the animal's neck during a chase and arrest, police said today. The dog, Rex, was on patrol with Constable Philip Rajah in the Natal provincial capital during the weekend when they came across two suspicious individuals," police said. While Rajah searched one man, Rex chased the other and got the worst of it when his quarry turned on the animal and bit him. Rajah had to yank the man off the dog, police said. They said the dog was being treated for a serious neck injury at a veterinary clinic. The man who bit the dog may face a charge of malicious injury to state property.

640 of 1000 characters used

Language Auto-Detect

Detected language: English

Analyze

## Entity

This API returns the named entities ("Person", "Organization", "Locations", etc.) within the text you analyzed.

List Tiles JSON

Filter Show all categories

**4-year-old**

# 0.96

Quantity Confidence

**today**

📅 0.97

Date Confidence

**Rex**

👤 0.96

Person Confidence

**Constable Philip ...**

👤 0.85

Person Confidence

**Natal**

📍 0.61

Location Confidence

**two suspicious in...**

# 0.89

Quantity Confidence

**Rajah**

👤 0.99

Person Confidence

**one man**

# 0.89

Quantity Confidence

<https://medium.com/@julsimon/a-quick-look-at-natural-language-processing-with-amazon-comprehend-238b8d9ec11d>

# Using the APIs Programmatically

- Of course each of these APIs can be used programmatically.
- Ex: Google Cloud Vision API

```
# Imports the Google Cloud client library
from google.cloud import vision
from google.cloud.vision import types

# Instantiates a client
client = vision.ImageAnnotatorClient()

# Performs label detection on the image file
response =
client.label_detection(image=image_filename)
labels = response.label_annotations
```



**Calling the Vision API in Python.** Now, calling the vision API in python. Can be the language of your choice!

# Some Other Popular ML APIs to look up

- IBM Watson
- Alchemy
- Google Prediction
- Wit.ai
- BigML

# ML frameworks

- ML Frameworks are more geared towards people (scientists) wanting to train and build their system from scratch.
- Some of the most popular ones include:
  - TensorFlow
  - PyTorch
  - Scikit-learn
  - Weka
  - Keras

PyTorch  
TensorFlow

# TensorFlow

- Google Brain's second-generation system.
- Can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions).
- Allows for the easy deployment of computation across platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.
- **Written in:** Python, C++, CUDA
- **Operating system:** Linux, macOS, Windows, Android, and iOS.



# Deep Neural Network for MNIST dataset classification task Using TensorFlow

```
from __future__ import division, print_function, absolute_import

import tflearn

# Data loading and preprocessing
import tflearn.datasets.mnist as mnist
X, Y, testX, testY = mnist.load_data(one_hot=True)

# Building deep neural network
input_layer = tflearn.input_data(shape=[None, 784])
dense1 = tflearn.fully_connected(input_layer, 64, activation='tanh',
                                  regularizer='L2', weight_decay=0.001)
dropout1 = tflearn.dropout(dense1, 0.8)
dense2 = tflearn.fully_connected(dropout1, 64, activation='tanh',
                                  regularizer='L2', weight_decay=0.001)
dropout2 = tflearn.dropout(dense2, 0.8)
softmax = tflearn.fully_connected(dropout2, 10, activation='softmax')

# Regression using SGD with learning rate decay and Top-3 accuracy
sgd = tflearn.SGD(learning_rate=0.1, lr_decay=0.96, decay_step=1000)
top_k = tflearn.metrics.Top_k(3)
net = tflearn.regression(softmax, optimizer=sgd, metric=top_k,
                          loss='categorical_crossentropy')

# Training
model = tflearn.DNN(net, tensorboard_verbose=0)
```

# PyTorch

- Is an open source machine learning library based on the Torch library
- Can be used for computer vision and natural language processing applications
- Is developed by Facebook's AI Research lab (FAIR).
- Is free and open-source software
- **Written in:** Python, C++
- **Operating system:** Linux, macOS, Windows





```
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
```

✓ 0.2s

PyTorch & TorchVision (Python) [PyTorch Install](#)

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 10, kernel_size=5)
        self.conv2 = nn.Conv2d(10, 20, kernel_size=5)
        self.conv2_drop = nn.Dropout2d()
        self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(50, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
        x = x.view(-1, 320)
        x = F.relu(self.fc1(x))
        x = F.dropout(x, training=self.training)
        x = self.fc2(x)
        return F.log_softmax(x)
```

✓ 0.2s

Creating the Network | PyTorch & TorchVision (Python) [PyTorch Install](#)

```
network = Net()
optimizer = optim.SGD(network.parameters(), lr=learning_rate,
                        momentum=momentum)
```

✓ 0.2s

Network & Optimizer Setup | PyTorch & TorchVision (Python) [PyTorch Install](#)

```
train_losses = []
train_counter = []
test_losses = []
test_counter = [i*len(train_loader.dataset) for i in range(n_epochs + 1)]
```

✓ 0.2s

PyTorch & TorchVision (Python) [PyTorch Install](#)

```
def train(epoch):
    network.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        optimizer.zero_grad()
        output = network(data)
        loss = F.nll_loss(output, target)
        loss.backward()
        optimizer.step()
    if batch_idx % log_interval == 0:
        print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
            epoch, batch_idx * len(data), len(train_loader.dataset),
              100. * batch_idx / len(train_loader), loss.item()))
        train_losses.append(loss.item())
        train_counter.append(
            (batch_idx*64) + ((epoch-1)*len(train_loader.dataset)))
        torch.save(network.state_dict(), '/results/model.pth')
        torch.save(optimizer.state_dict(), '/results/optimizer.pth')
```

# Deep Neural Network for MNIST dataset classification task Using PyTorch

# Scikit-learn (also known as sklearn)

- Is a free machine learning library for Python.
- Performs classification, regression, and clustering algorithms, including support vector machines, random forests, etc.
- Interoperate with NumPy and SciPy.
- Started as scikits.learn, a Google Summer of Code project by David Cournapeau.
- **Written in:** Python, Cython, C, C++
- **Operating system:** Linux, macOS, Windows



# MNIST classification by multinomial logistic + L1 Using Scikit-learn

```
import time
import matplotlib.pyplot as plt
import numpy as np

from sklearn.datasets import fetch_openml
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.utils import check_random_state

print(__doc__)

# Author: Arthur Mensch <arthur.mensch@m4x.org>
# License: BSD 3 clause

# Turn down for faster convergence
t0 = time.time()
train_samples = 5000

# Load data from https://www.openml.org/d/554
X, y = fetch_openml('mnist_784', version=1, return_X_y=True)

random_state = check_random_state(0)
permutation = random_state.permutation(X.shape[0])
X = X[permutation]
y = y[permutation]
X = X.reshape((X.shape[0], -1))

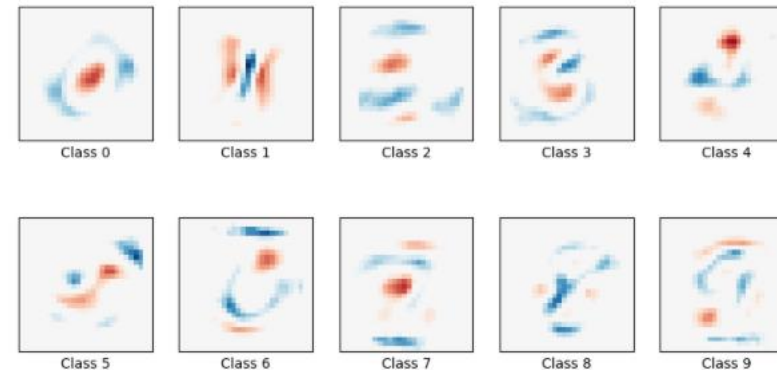
X_train, X_test, y_train, y_test = train_test_split(
    X, y, train_size=train_samples, test_size=10000)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Turn up tolerance for faster convergence
clf = LogisticRegression(
    C=50. / train_samples, penalty='l1', solver='saga', tol=0.1
)
clf.fit(X_train, y_train)
sparsity = np.mean(clf.coef_ == 0) * 100
score = clf.score(X_test, y_test)
# print('Best C % .4f' % clf.C_)
print("Sparsity with L1 penalty: %.2f%%" % sparsity)
print("Test score with L1 penalty: %.4f" % score)

coef = clf.coef_.copy()
plt.figure(figsize=(10, 5))
scale = np.abs(coef).max()
for i in range(10):
    l1_plot = plt.subplot(2, 5, i + 1)
    l1_plot.imshow(coef[i].reshape(28, 28), interpolation='nearest',
                  cmap=plt.cm.RdBu, vmin=-scale, vmax=scale)
    l1_plot.set_xticks(())
    l1_plot.set_yticks(())
    l1_plot.set_xlabel('Class %i' % i)
plt.suptitle('Classification vector for...')

run_time = time.time() - t0
print('Example run in %.3f s' % run_time)
plt.show()
```



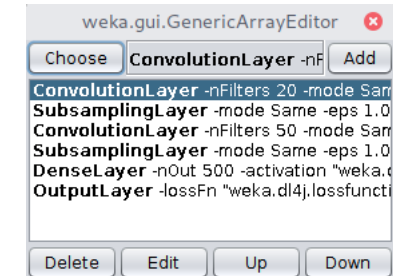
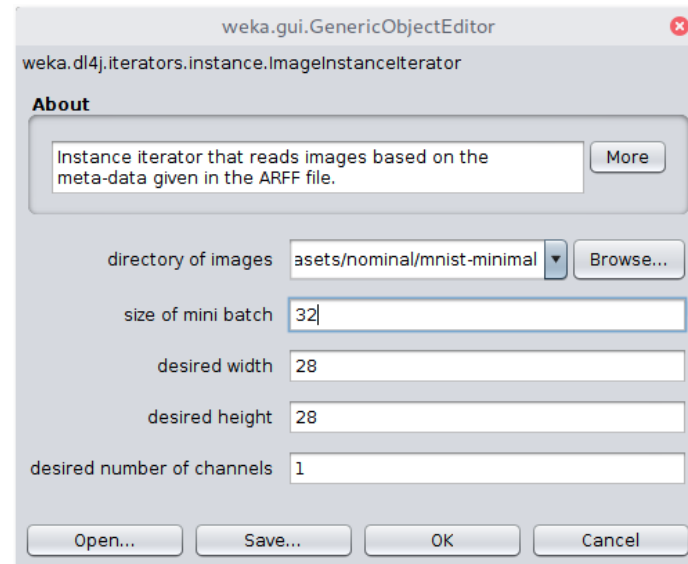
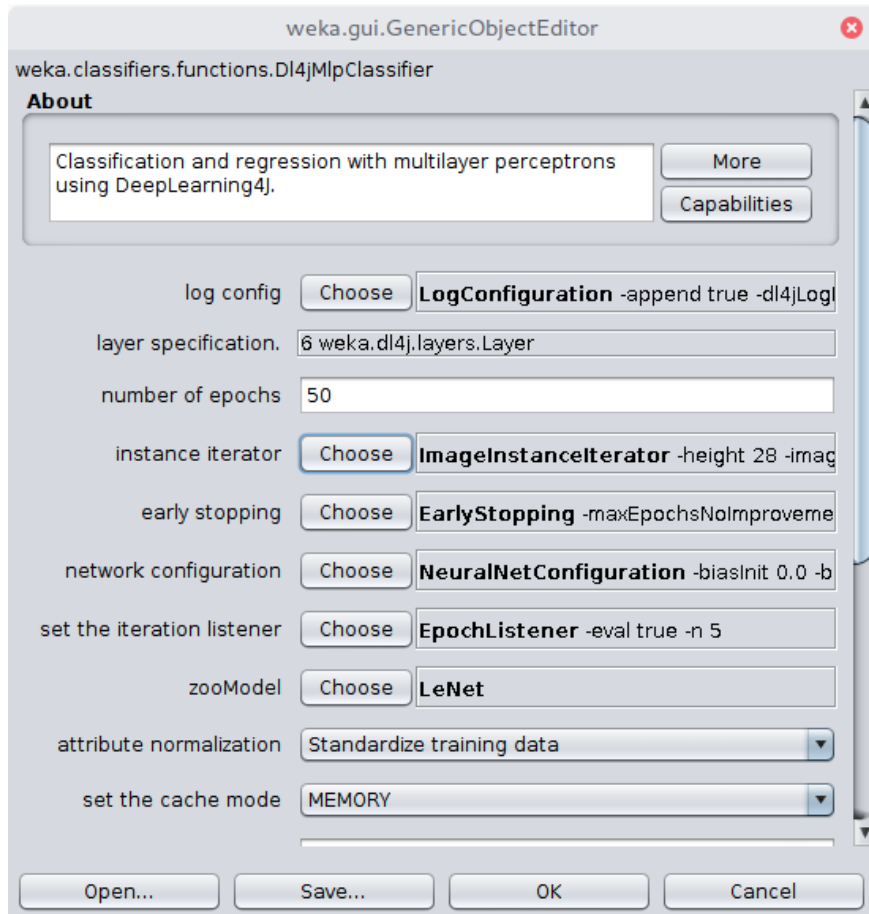
Ref: [https://scikit-learn.org/stable/auto\\_examples/linear\\_model/plot\\_sparse\\_logistic\\_regression\\_mnist.html?highlight=mnist](https://scikit-learn.org/stable/auto_examples/linear_model/plot_sparse_logistic_regression_mnist.html?highlight=mnist)

# Weka

- **Weka** is an open source machine learning software
- Can be accessed through a GUI, terminal, or a Java API.
- Contains built-in tools for standard machine learning tasks
- Gives transparent access to well-known toolboxes such as scikit-learn, R, and Deeplearning4j.
- **Written in:** Java
- **Operating system:** Linux, OS X, Windows



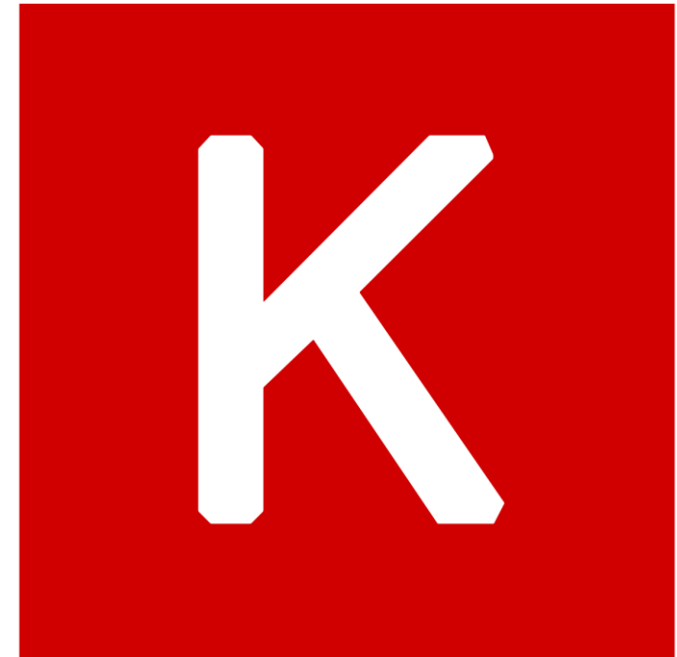
# Classifying MNIST dataset using Weka



<https://deeplearning.cms.waikato.ac.nz/examples/classifying-mnist/>

# Keras

- **Keras** is an open-source neural-network library written in Python.
- Can run on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, and PlaidML.
- Enables fast experimentation with deep neural networks
- Focuses on being user-friendly, modular, and extensible.
- **Written in:** Python
- **Operating system:** Cross-platform



# Deep Neural Network for MNIST dataset classification task Using Keras

[https://keras.io/examples/mnist\\_cnn/](https://keras.io/examples/mnist_cnn/)

```
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

# References

- <https://blog.rapidapi.com/top-machine-learning-apis/>
- <https://blog.exxactcorp.com/tensorflow-vs-pytorch-vs-keras-for-nlp/>
- <https://towardsdatascience.com/the-most-useful-ml-tools-2020-e41b54061c58>
- <https://en.wikipedia.org/wiki/>