

# MACHINE LEARNING DESDE CERO CON PYTHON HASTA EL DESPLIEGUE EN GOOGLE CLOUD



# Juan Guillermo Gómez

<vanity>



- Co-Leader y Co-Founder del GDG Cali.
- CEO DevHack.
- Consultant and advisor on software architecture, cloud computing and software development.
- Experience in several languages and platforms. (C, C#, Java, NodeJS, android, GCP, Firebase).
- Google Developer Expert (GDE) in Firebase
- BS in System Engineering and a MS in Software Engineering.
- @jggomez
- jggomez@devhack.co

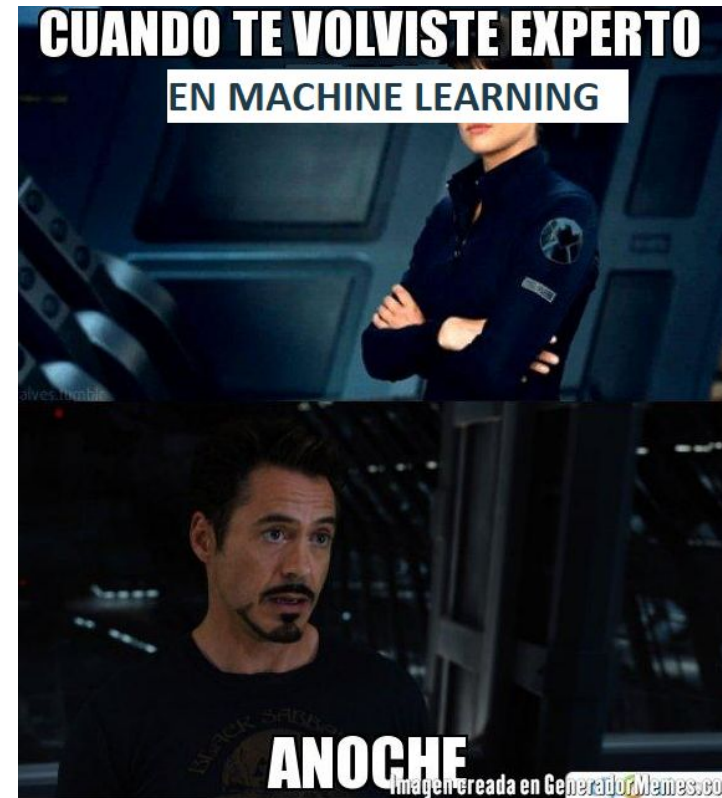


</vanity>



# Aprendizaje Continuo

- Autodidacta.
- Asiste a eventos
- Investiga.
- Práctica.
- Coding.
- Comparte
- Pierde el miedo.



# Machine Learning

- Learning is any process by which a system improves performance from experience (data).
- Machine Learning is concerned with computer programs that automatically improve their performance (predictions) through experience.

# Hebert Alexander Simon



# Why now?

- Flood of available data.
- Increasing computational power.
- Growing progress in available algorithms and theory developed by researchers.
- Increasing support from industries.
- Machine Learning is concerned with computer programs that automatically improve their performance through experience.



# ML Applications





# Where I Start?

## APIs



### Cloud Vision API

#### Powerful image analysis

Cloud Vision API enables you to derive insight from your images with our powerful pretrained API models or easily train custom vision models with AutoML Vision Beta. The API quickly classifies images into thousands of categories (such as “sailboat” or “Eiffel Tower”), detects individual objects and faces within images, and finds and reads printed words contained within images. AutoML Vision lets you build and train custom ML models with minimal ML expertise to meet domain-specific business needs.



### Cloud Speech-to-Text

#### Speech recognition across 120 languages

Cloud Speech-to-Text enables developers to convert audio to text by applying neural network models in an easy-to-use API. The API recognizes 120 languages and variants, to support your global user base. You can enable voice command-and-control, transcribe audio from call centers, and more. It can process real-time streaming or prerecorded audio, using Google’s machine learning technology.

[LEARN MORE ABOUT CLOUD SPEECH-TO-TEXT](#)

# Where I Start?

## APIs



### Cloud Natural Language API

#### Multimedia and multi-language processing

Cloud Natural Language API reveals the structure and meaning of text by offering powerful machine learning models in an easy-to-use REST API. And with AutoML Natural Language Beta you can build and train ML models easily, without extensive ML expertise. You can use Natural Language to extract information about people, places, events, and much more mentioned in text documents, news articles, or blog posts. You can also use it to understand sentiment about your product on social media or parse intent from customer conversations happening in a call center or a messaging app.

[LEARN MORE ABOUT CLOUD NATURAL LANGUAGE](#)



### Cloud Video Intelligence API

#### Precise video analysis — down to the frame

Cloud Video Intelligence API makes videos searchable and discoverable by extracting metadata, identifying key nouns, and annotating the content of the video. By calling an easy-to-use REST API, you can now search every moment of every video file in your catalog and find each occurrence of key nouns as well as its significance. Separate signal from noise by retrieving relevant information by video, shot, or frame.

[LEARN MORE ABOUT CLOUD VIDEO INTELLIGENCE API](#)



# Where I Start?

## APIs



### Cloud AutoML

High-quality custom machine learning models

Cloud AutoML Beta is a suite of machine learning products that enables developers with limited machine learning expertise to train high-quality models specific to their business needs by leveraging Google's state-of-the-art transfer learning and neural architecture search technology.

[LEARN MORE ABOUT CLOUD AUTOML](#)

# Where I Start?

## APIs



### Cloud AutoML

High-quality custom machine learning models

Cloud AutoML Beta is a suite of machine learning products that enables developers with limited machine learning expertise to train high-quality models specific to their business needs by leveraging Google's state-of-the-art transfer learning and neural architecture search technology.

[LEARN MORE ABOUT CLOUD AUTOML](#)

# Where I Start?

## Improve your contact center with AI

Designed to work with existing contact center technology, Contact Center AI makes it easy to train an AI model to interact with customers and provide insightful direction for agents. The result? A more personalized, intuitive customer care experience from the first "Hello."



## Smarter agent tools and at-the-ready resources

When calls are forwarded to a live agent, Agent Assist presents machine-learning-driven insights, helping the agent provide personalized and relevant upsells.

## Faster insights into your customer data

Bring the best of Google AI to every call. Improve customer service experience in your contact center through call automation, AI-powered assistance to human agents, and powerful analytics for business analysts and managers.

# Where I Start?

AutoML Vision **BETA** | jggomez | + ADD IMAGES | LABEL STATS | EXPORT DATA | appecommerceworkshop

IMAGES | TRAIN | EVALUATE | PREDICT

All images 69  
Labeled 69  
Unlabeled 0

Type to filter...

Select all images

juan 69  
Add label

Type to filter images...

juan juan juan juan juan juan juan

juan juan juan juan juan juan juan



# Where I Start?

AutoML Vision **BETA** | jggomez | + ADD IMAGES | LABEL STATS | EXPORT DATA | appecommerceworkshop

IMAGES | TRAIN | EVALUATE | PREDICT

All images 75  
Labeled 75  
Unlabeled 0

Type to filter images...

Select all images

Type to filter...

adry	6
juan	69

Add label

adry adry adry adry adry adry juan

juan juan juan juan juan juan juan



# Where I Start?

The screenshot displays the AutoML Vision interface. The top navigation bar includes the 'AutoML Vision BETA' logo, the user 'jggomez', and buttons for 'ADD IMAGES', 'LABEL STATS', and 'EXPORT DATA'. The main interface has tabs for 'IMAGES', 'TRAIN', 'EVALUATE', and 'PREDICT'. The 'TRAIN' tab is active, showing a message: 'Try labeling more images before training. Each label should have at least 100 images assigned. Fewer images result in lower precision and recall scores. [Learn more](#)'. Below this, there are progress bars for labels 'adry' and 'juan'. A 'START TRAINING' button is visible. A modal dialog box titled 'Train new model' is open, containing a 'Model name' field with the text 'jggomez\_v20181027061009', a paragraph about accuracy and training budget, a 'Training budget' slider set to '1 compute hour (free\*)', a 'Data summary' section showing '79 labeled images, 2 labels', and a footnote about the free trial. The dialog has 'CANCEL' and 'START TRAINING' buttons at the bottom.

AutoML Vision BETA jggomez + ADD IMAGES LABEL STATS EXPORT DATA appecommerceworkshop

IMAGES TRAIN EVALUATE PREDICT

Try labeling more images before training

Each label should have at least 100 images assigned. Fewer images result in lower precision and recall scores. [Learn more](#)

100  
0 500

adry

juan

Your images will be automatically split into [training and test sets](#) to evaluate the model's performance. Unlabeled images will not be used.

START TRAINING

Train new model

Model name  
jggomez\_v20181027061009

The accuracy of your model generally depends on how long you allow it to train, and the quality of your dataset. To train your model for more than one compute hour, your dataset needs at least 1000 labeled images.

Training budget

1 compute hour (free\*)


Data summary

79 labeled images, 2 labels


\* Your first compute hour is free, for up to 10 models each month. [Pricing guide](#)


CANCEL START TRAINING


# Where I Start?

 AutoML Vision BETA

jggomez

 ADD IMAGES ▾

 LABEL STATS

 EXPORT DATA

appecommerceworkshop ▾

IMAGES **TRAIN** EVALUATE PREDICT

Models [TRAIN NEW MODEL](#)

jggomez\_v20181027061009

Created



Oct 27, 2018

1 compute hour



Analyzed

79 images



2 labels, 8 test images

 Avg precision 

1.0


 Precision 

100.0%

 Recall 


100.0%




Precision and recall are based on a score threshold of 0.5



[SEE FULL EVALUATION](#) [RESUME TRAINING](#) 

⋮

# Where I Start?


 AutoML Vision BETA

jggomez  ADD IMAGES  LABEL STATS  EXPORT DATA

### Test your model on new images

If your model will be used to make predictions on people, test your model on images that capture the diversity of your userbase. [Learn more](#)




×

Predictions



Only top 3 labels are shown.


juan	<div></div>	1.000
adry	<div></div>	0.000
--other--	<div></div>	0.000


# Where I Start?

 AutoML Vision BETA

jggomez + ADD IMAGES ▾ || LABEL STATS ⌘ EXPORT DATA

If your model will be used to make predictions on people, test your model on images that capture the diversity of your userbase. [Learn more](#) 



×

Predictions

Only top 3 labels are shown.

juan	<div></div>	1.000
adry	<div></div>	0.000
--other--	<div></div>	0.000

# Where I Start?

REST API

PYTHON

predict.py

```
import sys

from google.cloud import automl_v1beta1
from google.cloud.automl_v1beta1.proto import service_pb2

def get_prediction(content, project_id, model_id):
    prediction_client = automl_v1beta1.PredictionServiceClient()

    name = 'projects/{}/locations/us-central1/models/{}'.format(project_id, model_id)
    payload = {'image': {'image_bytes': content }}
    params = {}
    request = prediction_client.predict(name, payload, params)
    return request # waits till request is returned

if __name__ == '__main__':
    file_path = sys.argv[1]
    project_id = sys.argv[2]
    model_id = sys.argv[3]

    with open(file_path, 'rb') as ff:
        content = ff.read()

    print get_prediction(content, project_id, model_id)
```



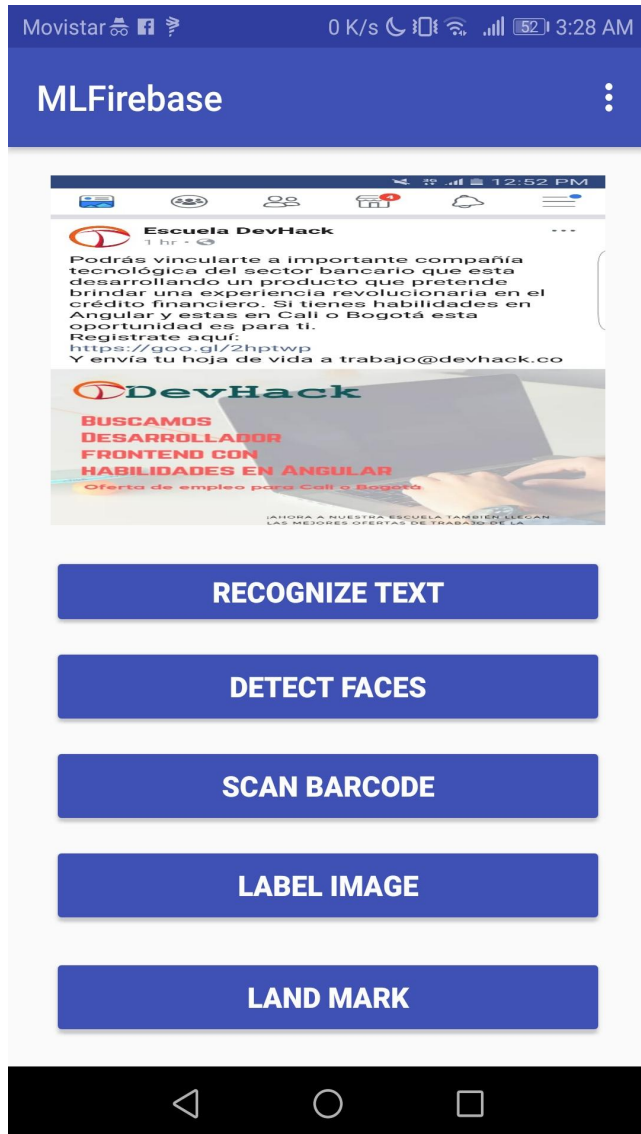
# Where I Start?

## ML Kit for Firebase

What features are available on device or in the cloud?

Feature	On-device	Cloud
Text recognition	✓	✓
Face detection	✓	
Barcode scanning	✓	
Image labeling	✓	✓
Landmark recognition		✓
Custom model inference	✓	

# Where I Start?



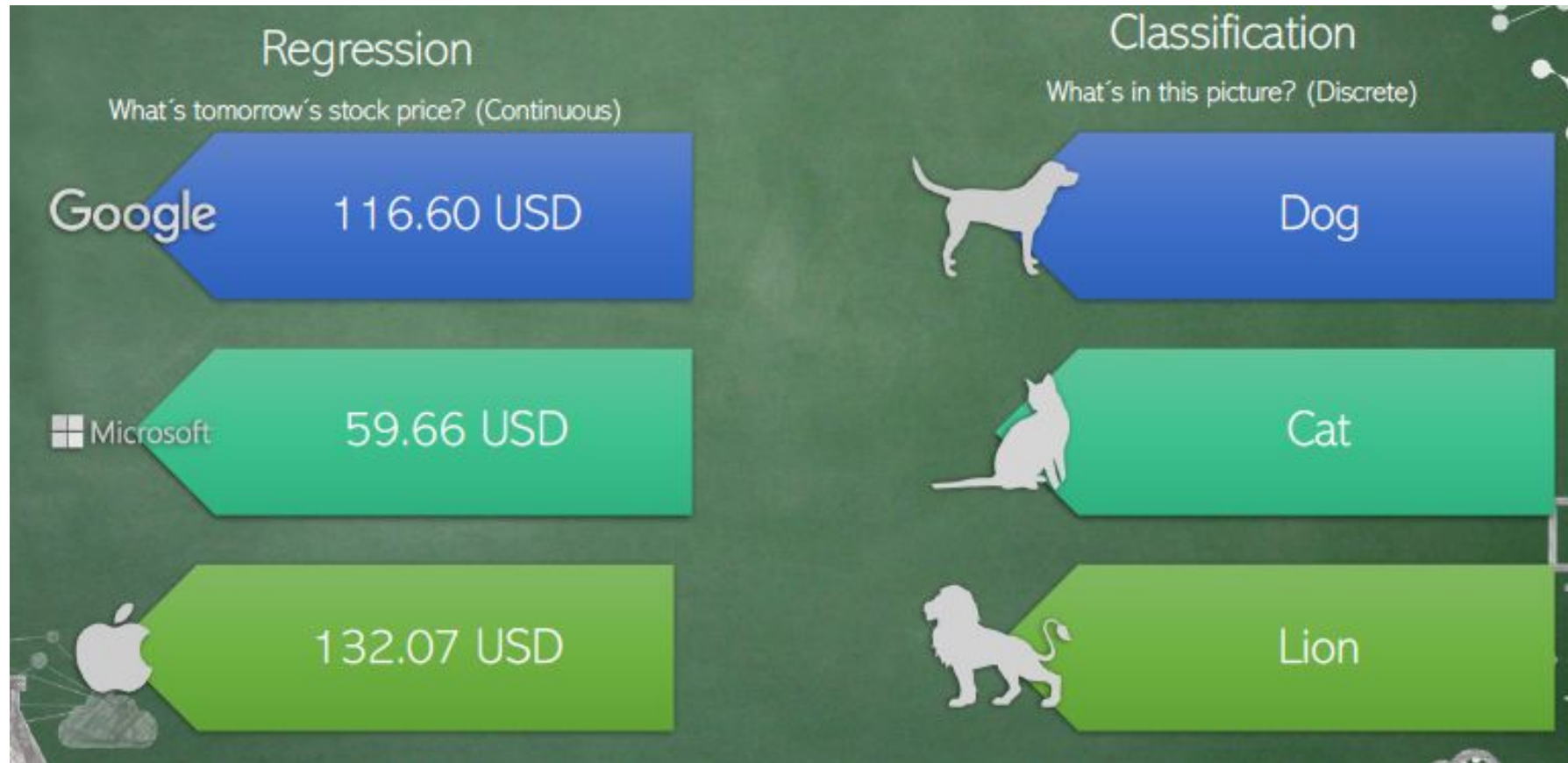
ML Kit for Firebase

[https://github.com/jggomez/ml\\_kit\\_firebase](https://github.com/jggomez/ml_kit_firebase)

# Model

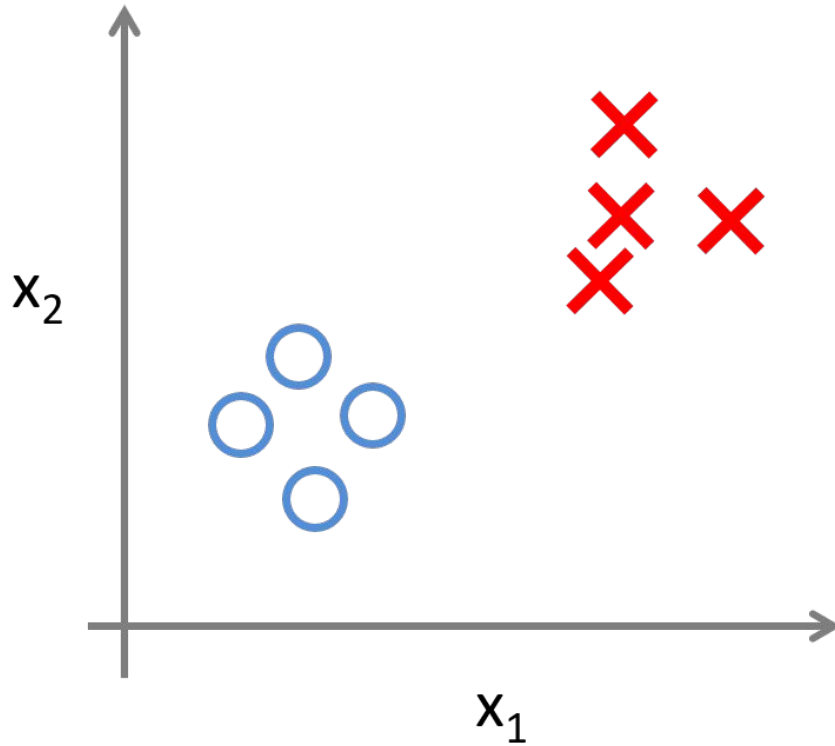


# Kind of Problems

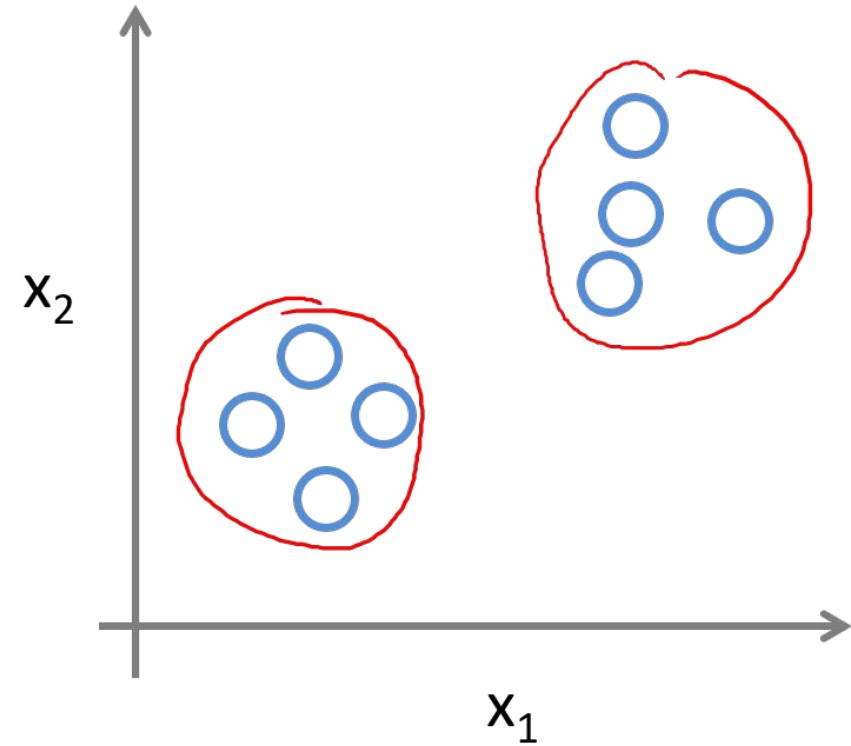


# Kind of Problems

Supervised Learning



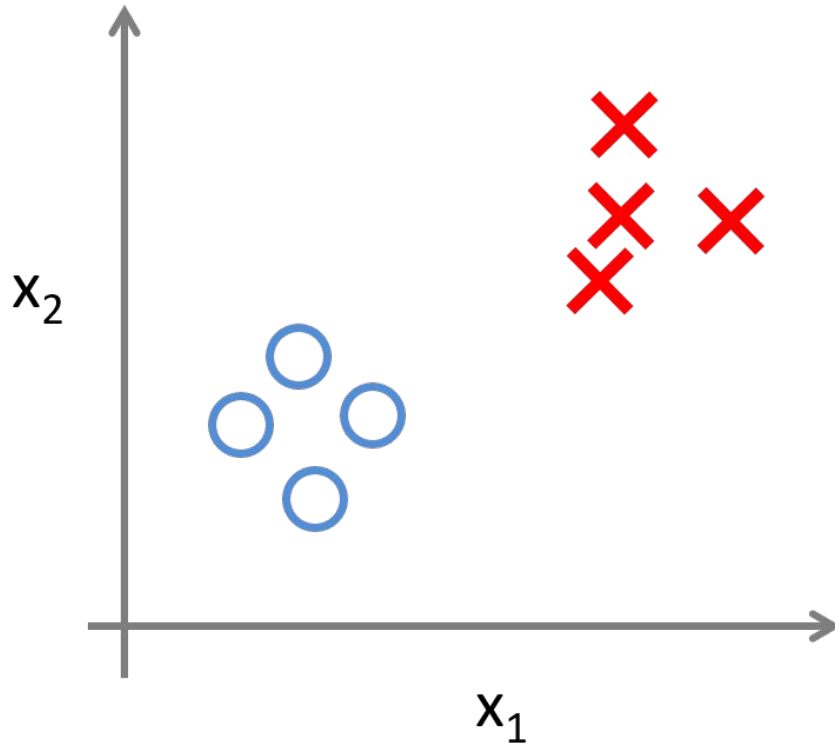
Unsupervised Learning



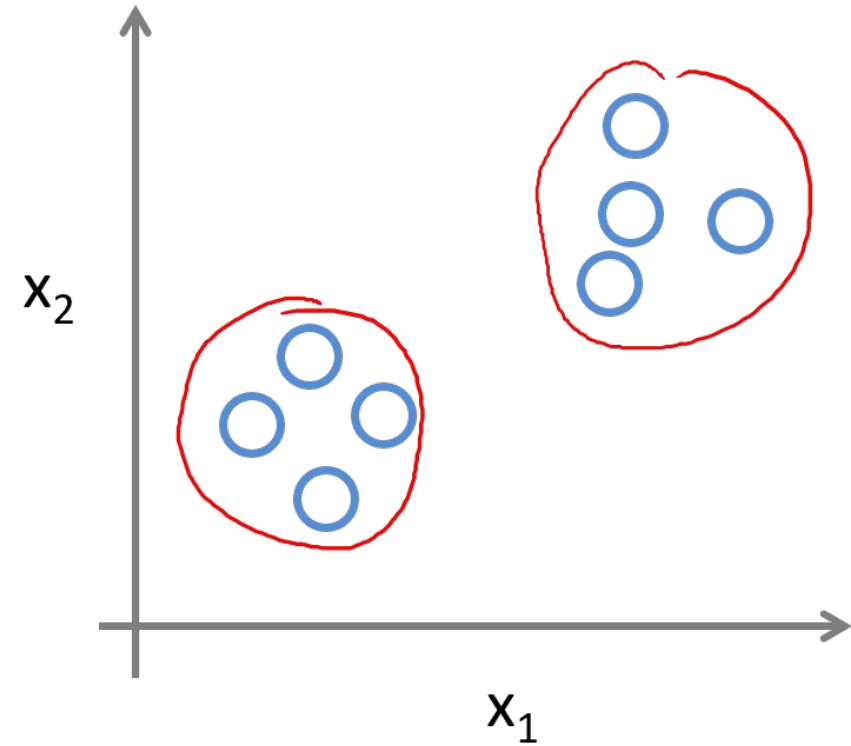


# Kind of Problems

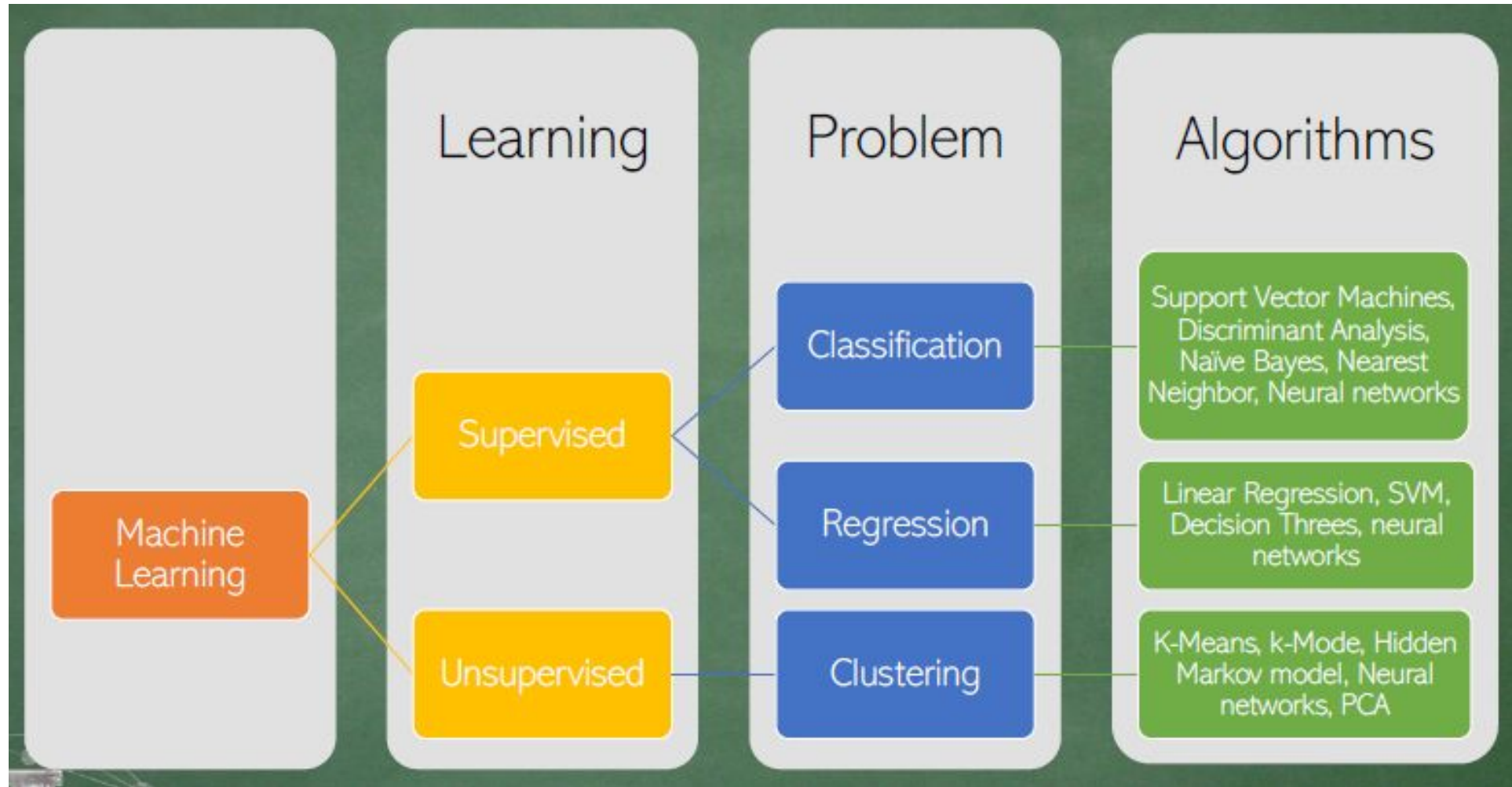
Supervised Learning



Unsupervised Learning



# Kind of Algorithms

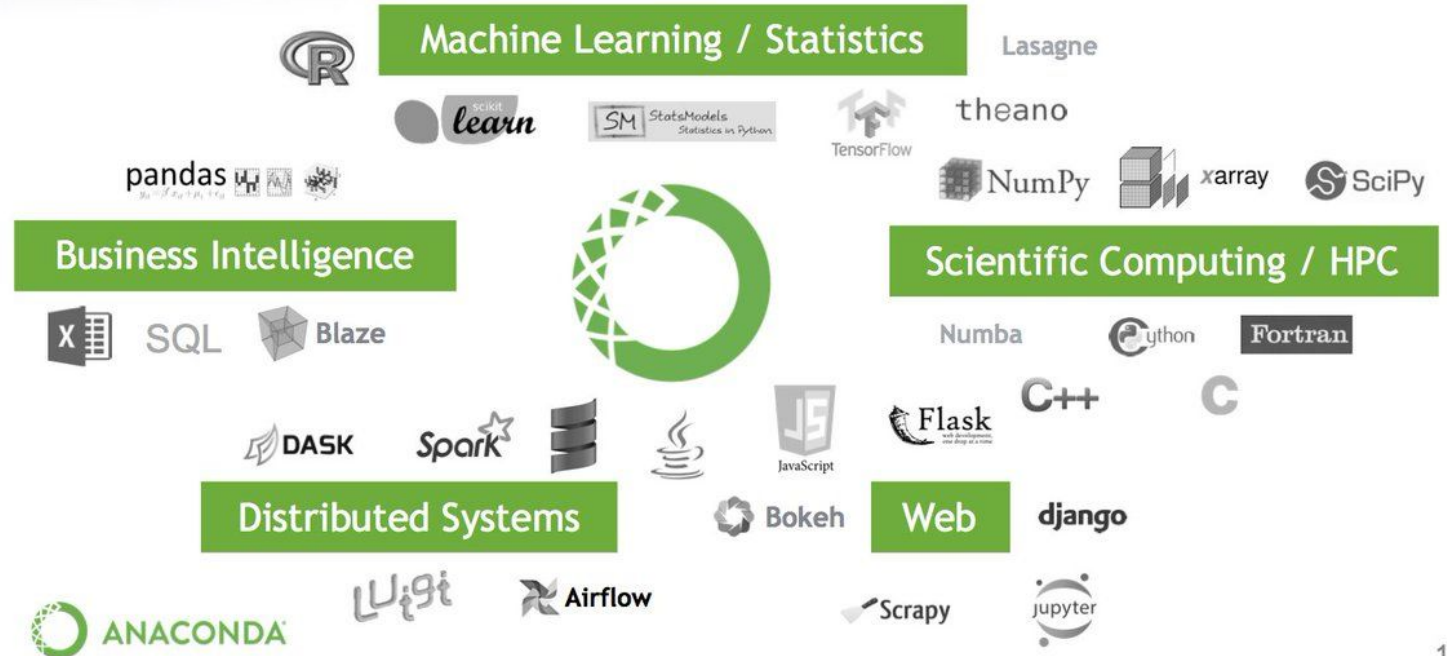




# Tools

Anaconda is the Open Data Science Platform Bringing Technology Together...

CONTINUUM<sup>®</sup>  
ANALYTICS



# Tools - IDE - DATA - Libs



Nombre

appleRed

banana

jupyter ClassifyFruits Last Checkpoint: 08/11/2018 (autosaved)



Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3

Run Code

```
In [31]: import cv2
import urllib.request as req
from matplotlib import pyplot as plt
import numpy as np
import glob
import os
import pandas # visualization & transformation
from pandas.tools.plotting import scatter_matrix
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
import pickle
import sklearn
print('The scikit-learn version is {}'.format(sklearn.__version__))
```

The scikit-learn version is 0.19.2.

evHack

# Tools - IDE - DATA - Libs

In [ ]:

In [8]: `DIR_DATA = "C:/Users/jggomez/Documents/MachineLearning/training_fruits"`

```
data = []
```

```
def getFeaturesExtractionByClass(name, classId):  
    classDir = os.path.join(DIR_DATA, name)  
    for imageName in glob.glob("{}/*.jpg".format(classDir)):  
        image = cv2.imread(imageName)  
        image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)  
        image = cv2.resize(image, (100, 100))  
        (means, stds) = cv2.meanStdDev(image)  
        stats = np.concatenate([means, stds])  
        stats = np.append(stats, classId)  
        data.append(stats.flatten())
```

```
getFeaturesExtractionByClass("Banana", 1)  
getFeaturesExtractionByClass("AppleRed", 2)  
#print(data)
```

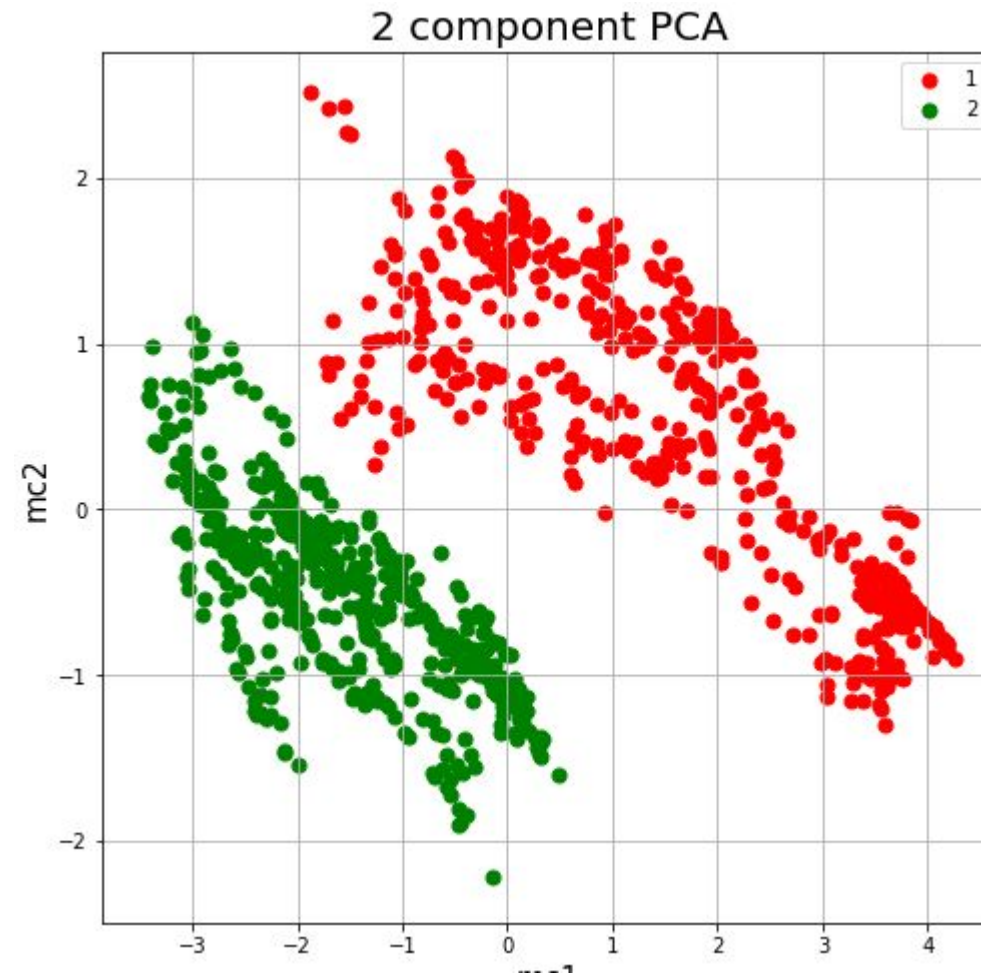


# Tools - IDE - DATA - Libs

```
In [10]: ## Standardize the Data
from sklearn.preprocessing import StandardScaler
features = ['c1', 'c2', 'c3', 'c4', 'c5', 'c6']
# Separating out the features
x = dataset.loc[:, features].values
# Separating out the target
y = dataset.loc[:, ['class']].values
# Standardizing the features
x = StandardScaler().fit_transform(x)
```

```
In [ ]: #PCA
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(x)
principalDf = pandas.DataFrame(data = principalComponents, columns = ['mc1', 'mc2'])
finalDf = pandas.concat([principalDf, dataset[['class']]], axis = 1)
```

# Tools - IDE - DATA - Libs



# Tools - IDE - DATA - Libs

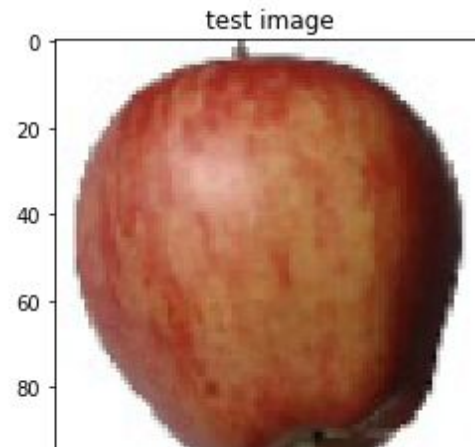
```
In [14]: # Spot Check Algorithms
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
# evaluate each model in turn
results = []
names = []
for name, model in models:
    →kfold = model_selection.KFold(n_splits=10, random_state=seed)
    →cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scoring)
    →results.append(cv_results)
    →names.append(name)
    →msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    →print(msg)
```

```
LR: 1.000000 (0.000000)
LDA: 1.000000 (0.000000)
KNN: 1.000000 (0.000000)
CART: 1.000000 (0.000000)
NB: 0.998734 (0.003797)
SVM: 0.989825 (0.015844)
```

# Tools - IDE - DATA - Libs

```
In [20]: import cv2
from matplotlib import pyplot as plt
path= "test_fruits/appletest.jpg"
image = cv2.imread(path)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
#cv2.imwrite(path, image)
plt.figure()
plt.title("test image")
plt.imshow(image)
print(image.shape)
newImage = processImage(path)
print(newImage)

(100, 100, 3)
[ 20.3894    108.7278    180.1649    39.09395052    78.31752107
  60.75763086]
```



# Tools - IDE - DATA - Libs

```
In [23]: knn = KNeighborsClassifier()
knn.fit(X_train, Y_train.ravel())
filename = 'finalized_fruits_model.sav'
# save the model to disk
pickle.dump(knn, open(filename, 'wb'))

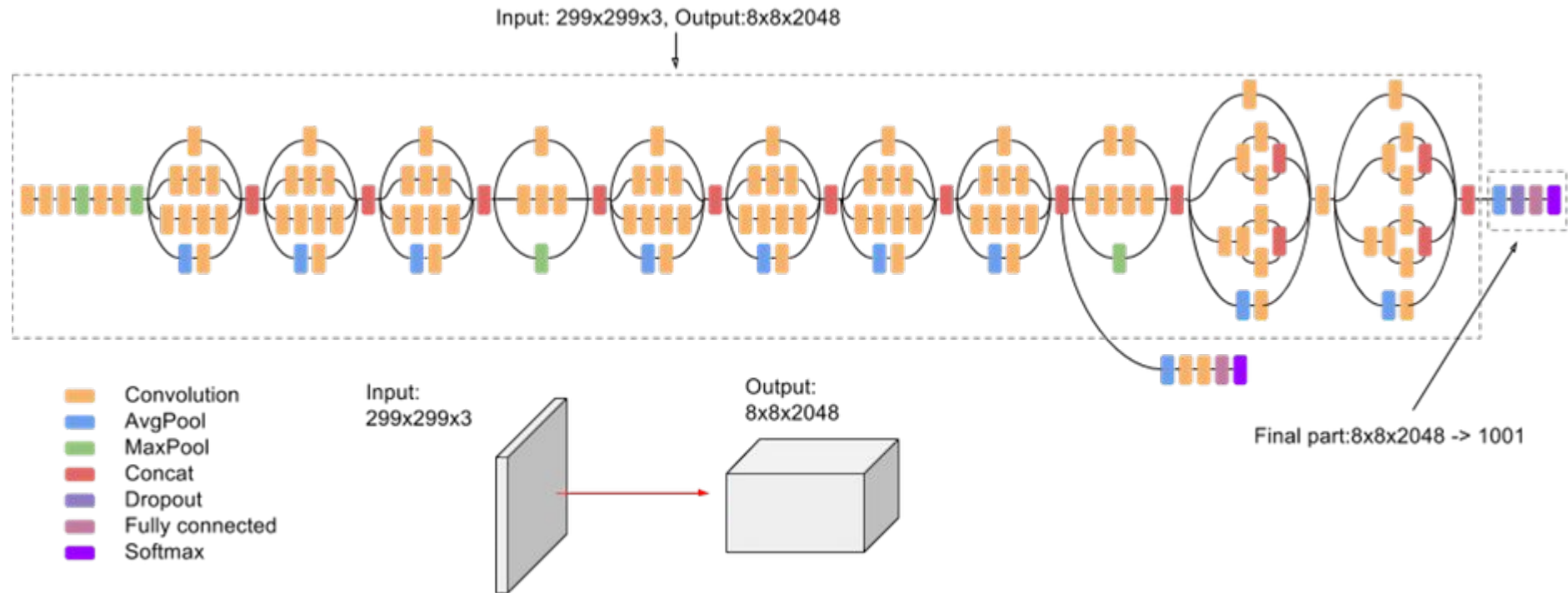
# load the model from disk
loaded_model = pickle.load(open(filename, 'rb'))
p = loaded_model.predict([newImage])
print(p)
```

[2.]



# Neural Network

## Inception V3 Architecture



```
python retrain.py --image_dir ~/flower_photos
```

# Neural Network

## Inception V3 Architecture

```
In [2]: !python label_image.py \
--graph="C:/tmp/output_graph.pb" \
--labels="C:/tmp/output_labels.txt" \
--input_layer=Placeholder \
--input_height=224 --input_width=224 \
--output_layer=final_result \
--image="dataset/appleRed/0_100.jpg"
```

applered 0.9995553

banana 0.000444695

# Deploy GCP

Google Cloud Platform

ML Engine

Tareas

Modelos

← Detalles del modelo

+ NUEVA VERSIÓN

OCULTAR PANEL DE INFORMACIÓN

Nombre

clasificacionFrutas

Versión predeterminada

-

Versiones

Filtrar por prefijo...

☐

Nombre

☐

v1

27 oct. 2018

4:04:12

Usada por última vez

Etiquetas

No se ha seleccionado ninguna versión

Las etiquetas ayudan a organizar los recursos (por ejemplo, centro\_costes:ventas o ent:prod).

No se ha seleccionado ninguna versión.



**Escuela de Hackers**  
**www.devhack.co**



**<https://www.facebook.com/escueladevhack/>**



**<https://github.com/escueladevhack>**