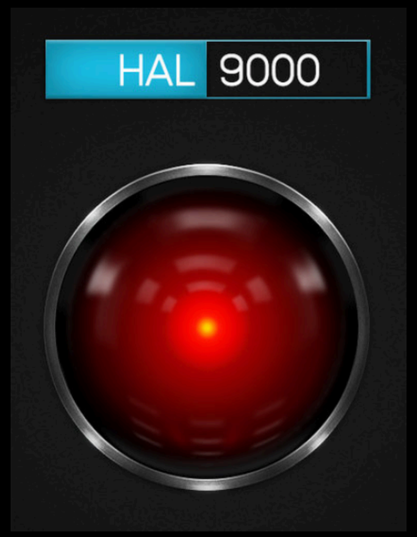
Beyond Speech Processing

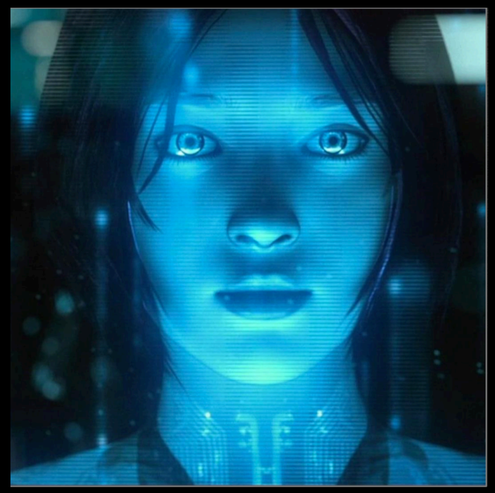
Toolkits for Accelerated Speech Processing and Machine Learning

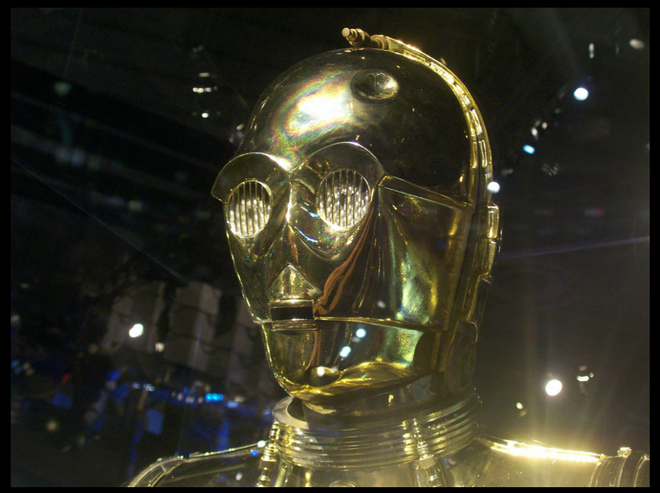
Trung Ngo Trong
trung@imito.ai

UNIVERSITY OF
EASTERN FINLAND

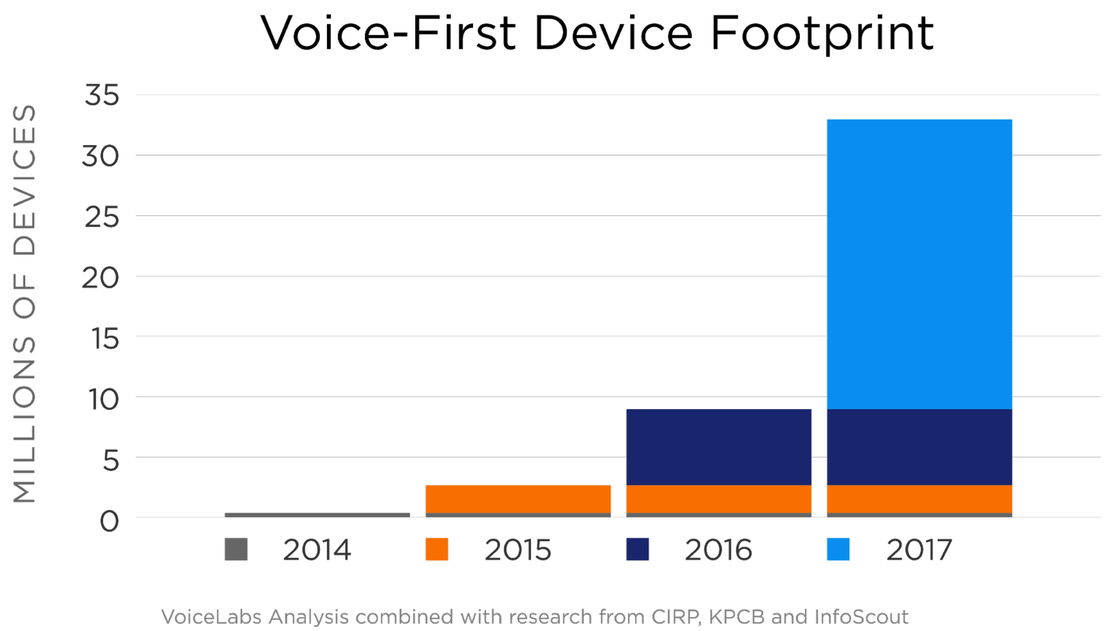
Computer Speech is the Dream



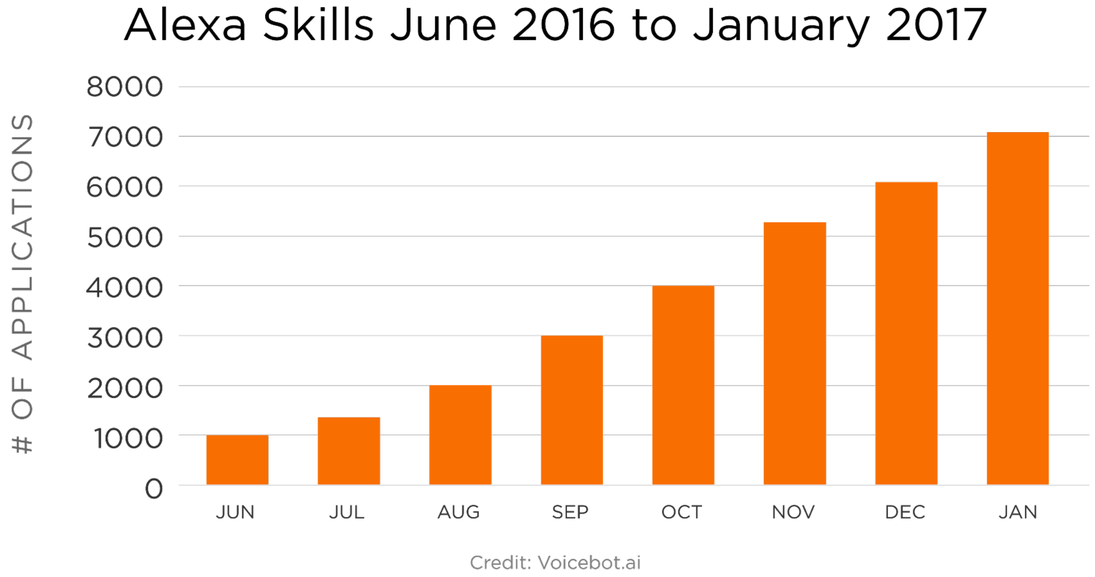




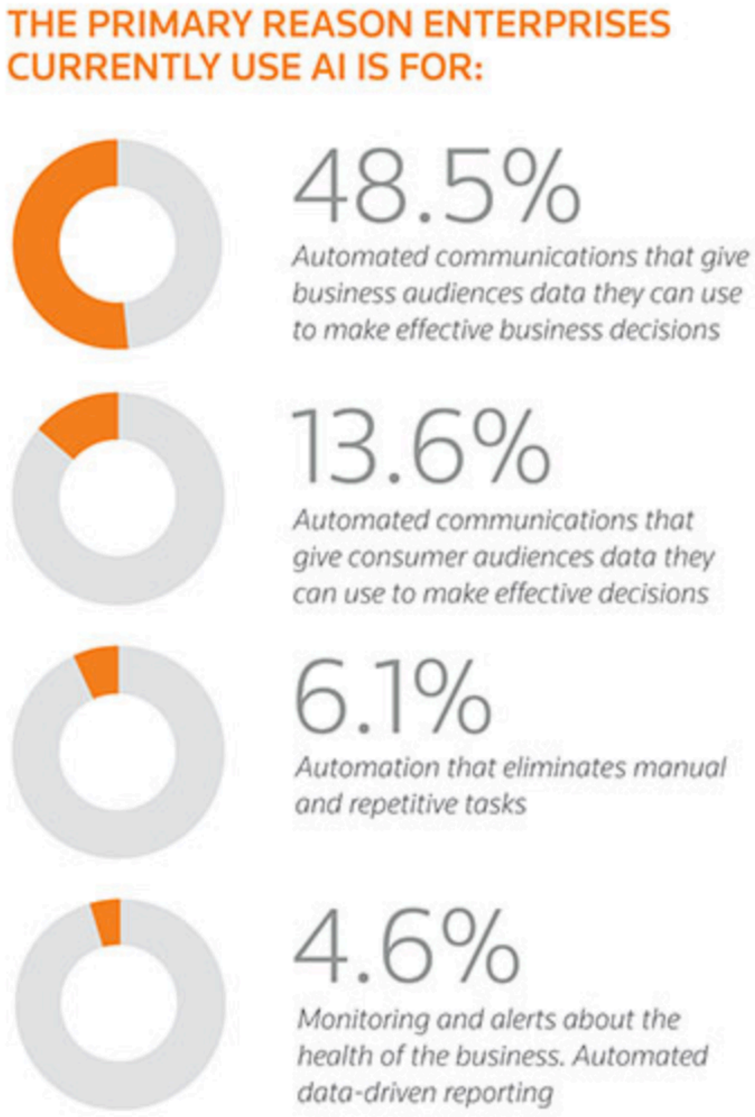
Industrial Response



VoiceLabs Analysis combined with research from CIRP, KPCB and InfoScout

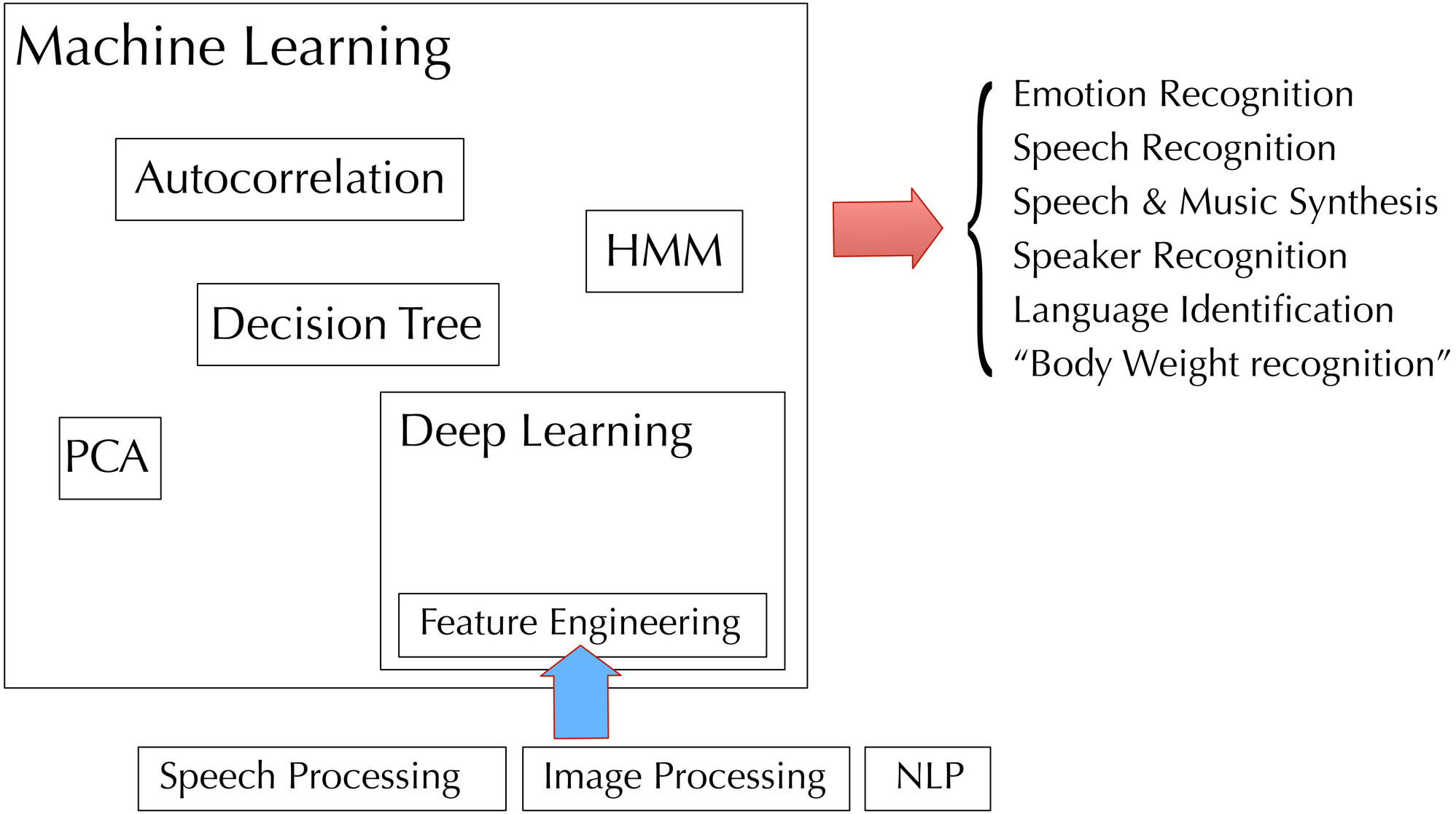


Credit: Voicebot.ai

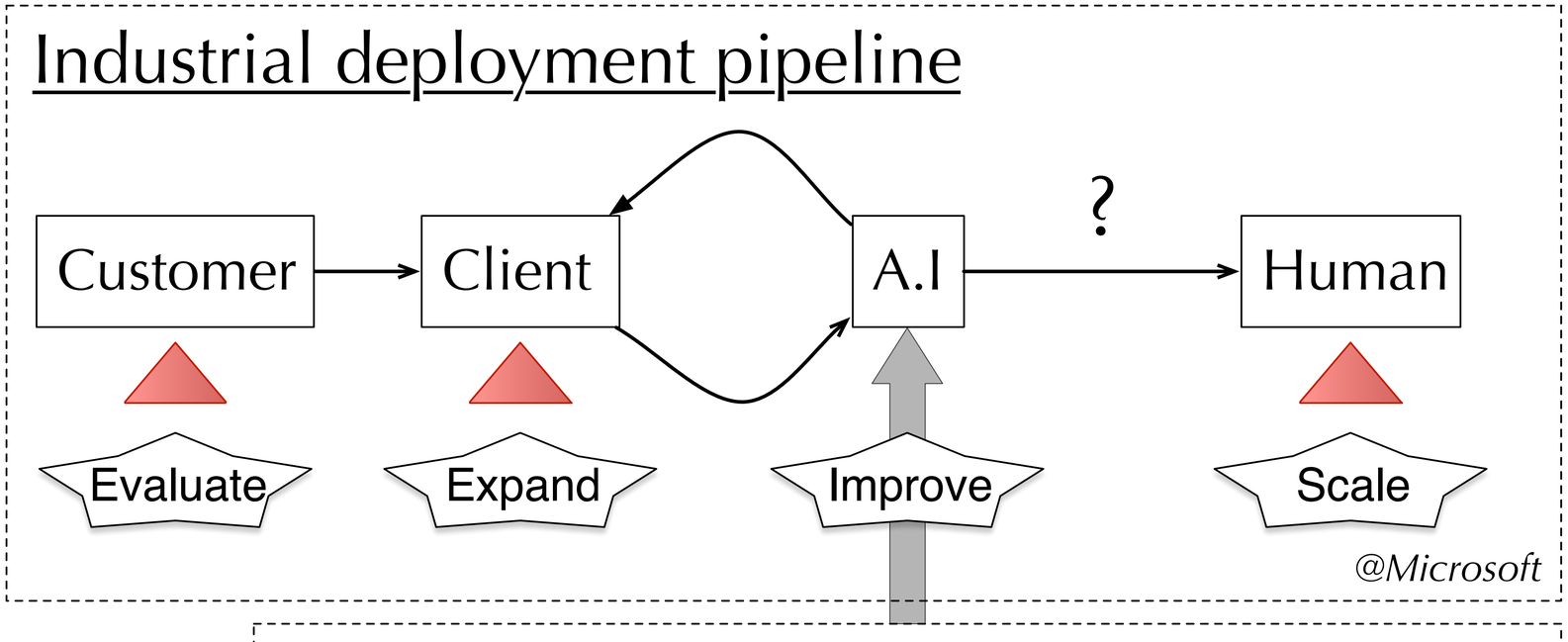


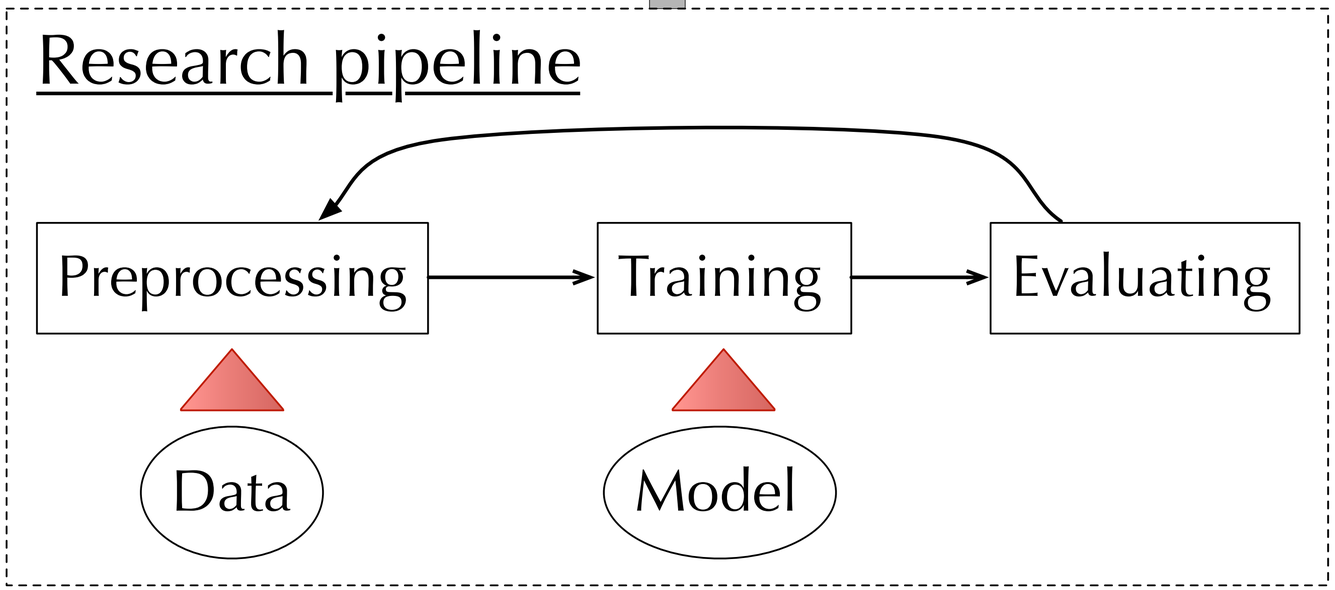
Statistics provided by: @campaignuk, @voicelabs, @microsoft, @reuters

Why Machine Learning !



The Pipeline





- Understand data collection is critical
- It is almost never work for the first time
- Speed iteration is more important than quality iteration

Best Languages ?

	Matlab	Python	R	C/C++
Setup and environment	✔✔	✔✔	✔✔✔	✔
Speed	✔✔	✔✔	✔✔	✔✔✔
Versatile	✔	✔✔✔	✔✔	✔✔
Syntax friendly	✔✔✔	✔✔✔	✔✔✔	✔
Development time	✔✔✔	✔✔	✔✔✔	✔
Libraries	✔✔	✔✔✔	✔✔	✔✔✔
Industrial application	✔	✔✔✔	✔✔✔	✔✔✔
Remark	Research standard, most of speech processing libraries initially built on Matlab	Industrial standard, Amazon, Google, Facebook, Microsoft …	Extremely easy and efficient for statistical inference	Fast, be able to access full features of ML libraries

Speech and Data Processing

Step 1

Step 2

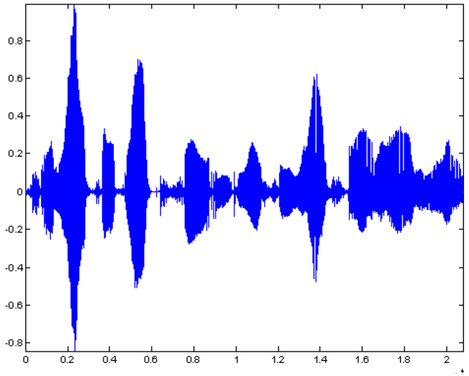
Step 3

Step 4

Step 5

Human

Machine

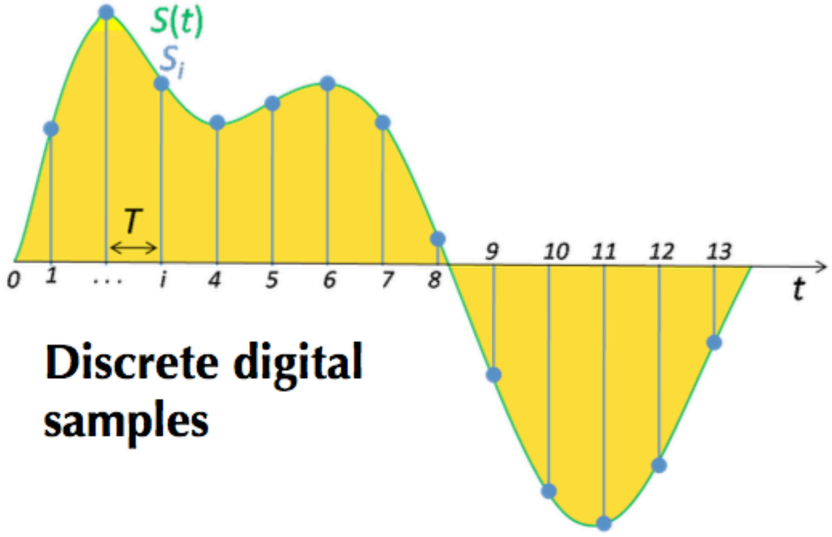


Sampling rate

Bit depth

Continuous sound wave

Miscrophone



Discrete digital samples

Speech and Data Processing

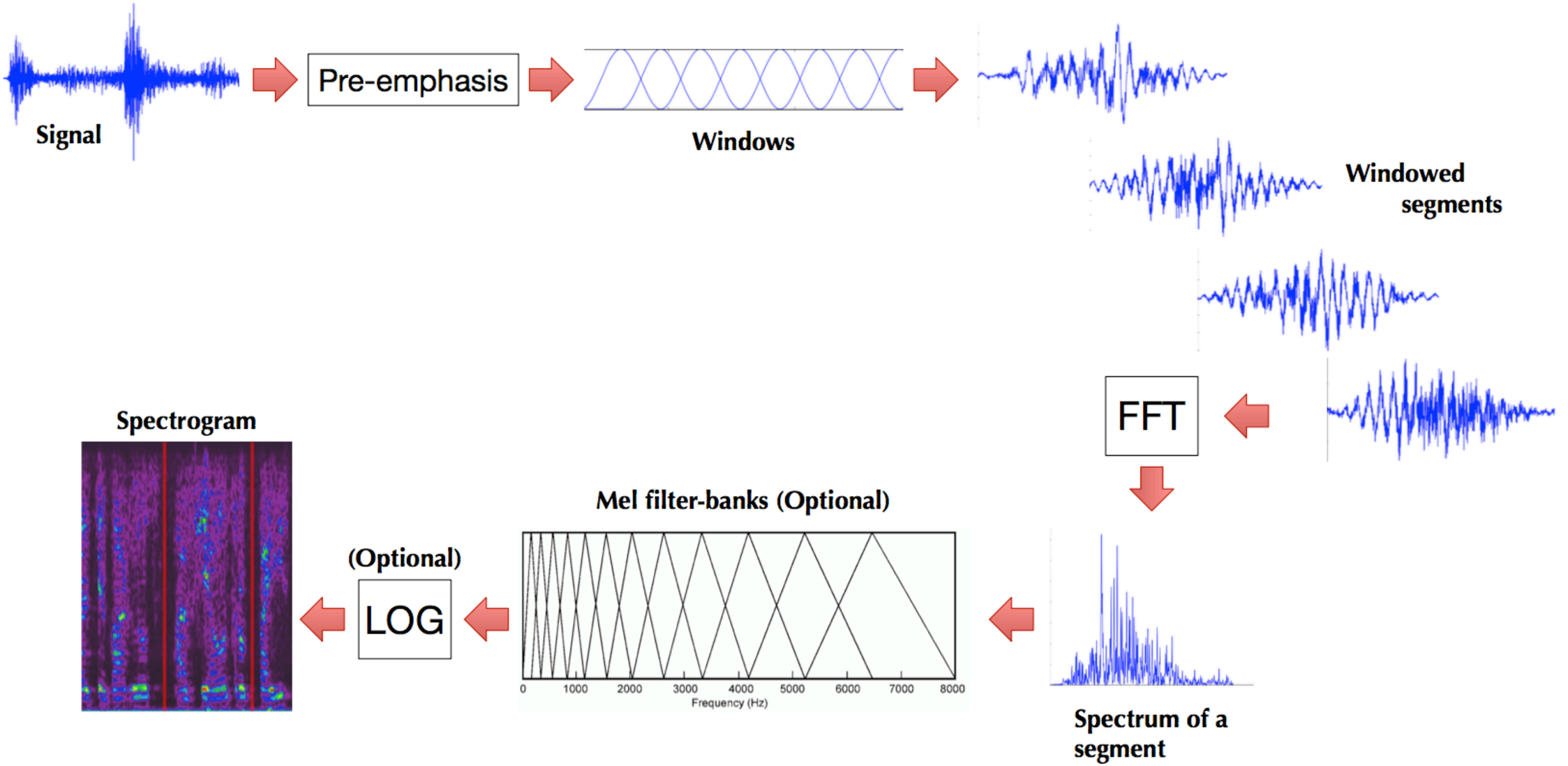
Step 1

Step 2

Step 3

Step 4

Step 5



Speech and Data Processing

Step 1

Step 2

Step 3

Step 4

Step 5

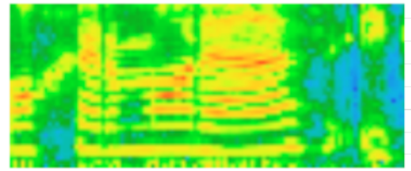
```python
import librosa
import numpy as np
y, sr = librosa.load('path/to/audio')
fft = librosa.stft(y, n_fft=256)
spectrogram = np.abs(fft) ** 2
mel_spec = librosa.feature.melspectrogram(S=spectrogram, sr=sr,
                                          n_mels=40, fmax=8000)
S = librosa.feature.mfcc(S=librosa.power_to_db(mel_spec), n_mfcc=13)
```

FFT spectrogram

129

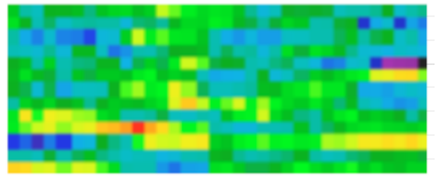
Mel-filterbank

40



MFCC

13



Libraries	sidekit	librosa	pysptk
Highlights	Speaker and Language recognition	Music and audio analysis	Speech analysis and synthesis (wrapper of C/C++ framework SPTK)

Speech and Data Processing

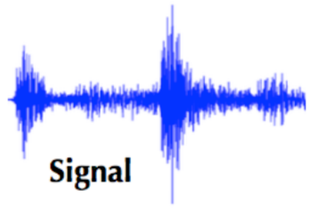
Step 1

Step 2

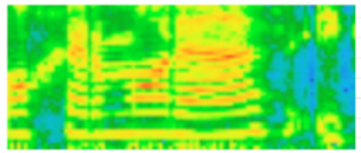
Step 3

Step 4

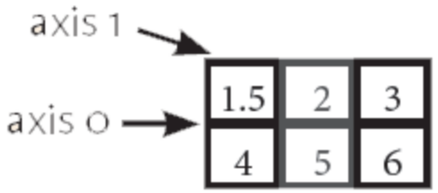
Step 5



1D array



2D array



Preprocessing

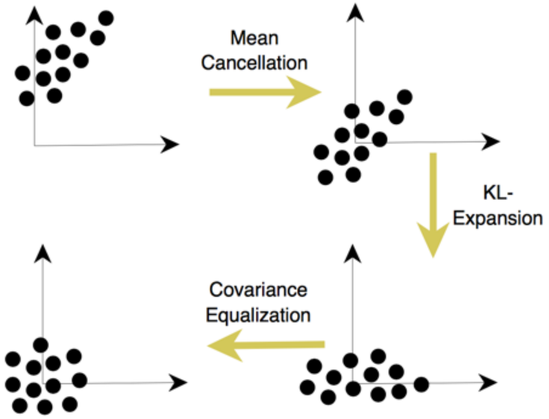
Data Managing

scikit learn

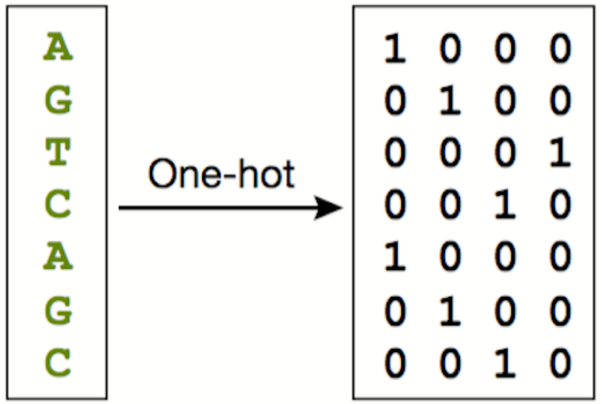
NumPy

SciPy

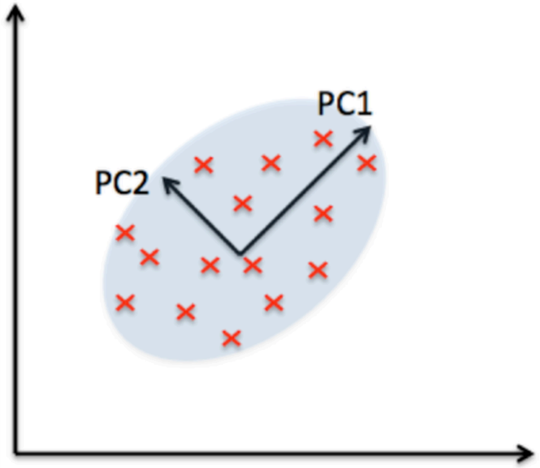
normalization



transformation



decomposition



Speech and Data Processing

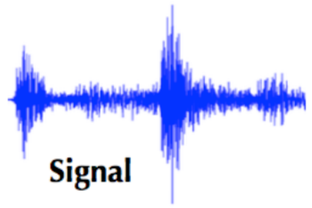
Step 1

Step 2

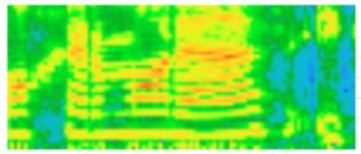
Step 3

Step 4

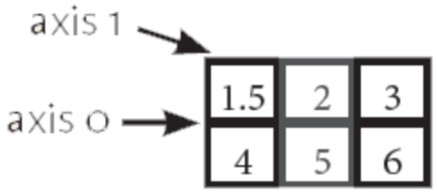
Step 5



1D array

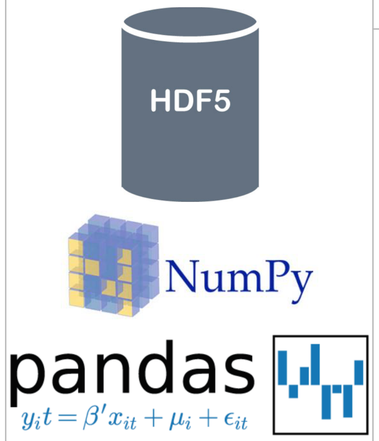


2D array

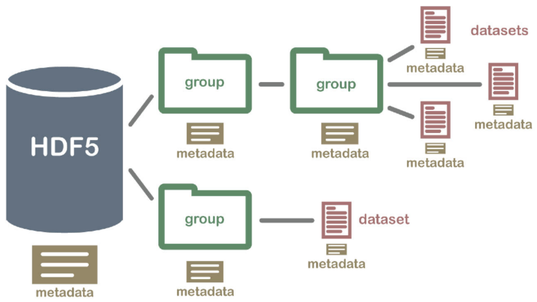


Preprocessing

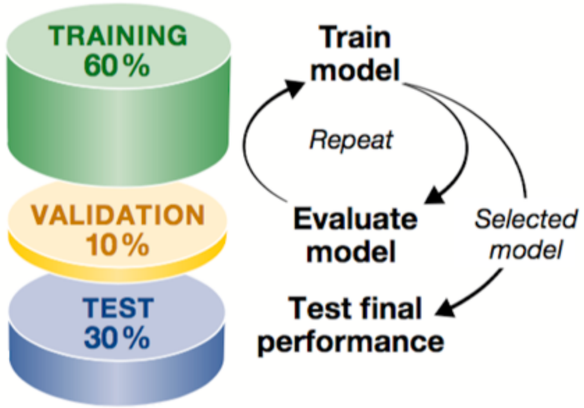
Data Managing



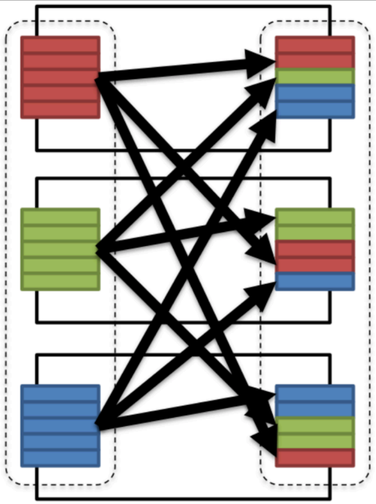
Storing the data



Splitting dataset



Training Schedule



Speech and Data Processing

Step 1

Step 2

Step 3

Step 4

Step 5

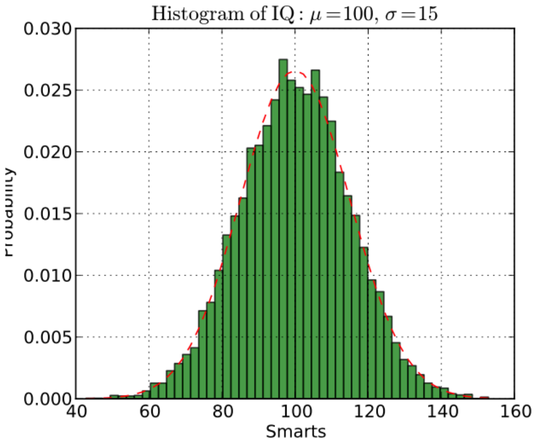
IP[y]:

```
In [9]: x = np.random.rand(3, 3)

In [10]: x
Out[10]:
array([[ 0.27881705,  0.01432253,  0.56128009],
       [ 0.18402877,  0.23905946,  0.23331274],
       [ 0.32774096,  0.35553655,  0.69962683]])

In [11]: np.random.rand?
Docstring:
rand(d0, d1, ..., dn)

Random values in a given shape.

Create an array of the given shape and populate it with
random samples from a uniform distribution
over ``[0, 1)``.
```

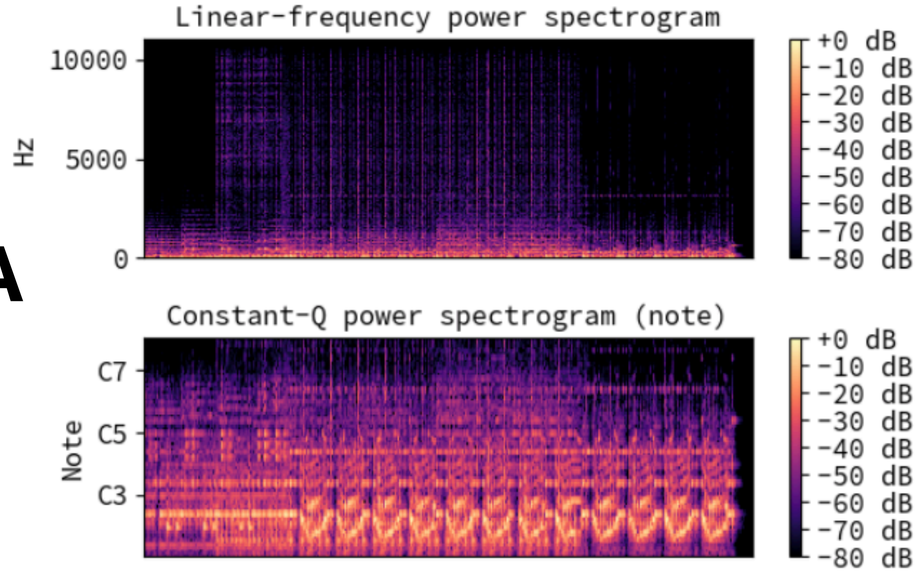
pandas

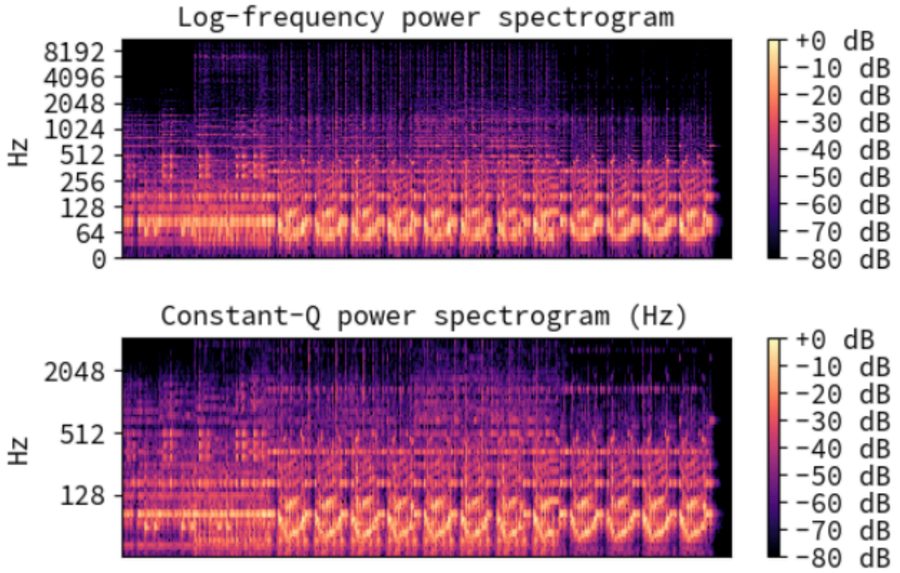
foo	one			two		
bar	Dog	Bird	Cat	Dog	Bird	Cat
2000-01-03	0	0	0	0	0	0
2000-01-04	0	0	0	0	0	0
2000-01-05	0	0	0	0	0	0

matplotlib



LIBROSA





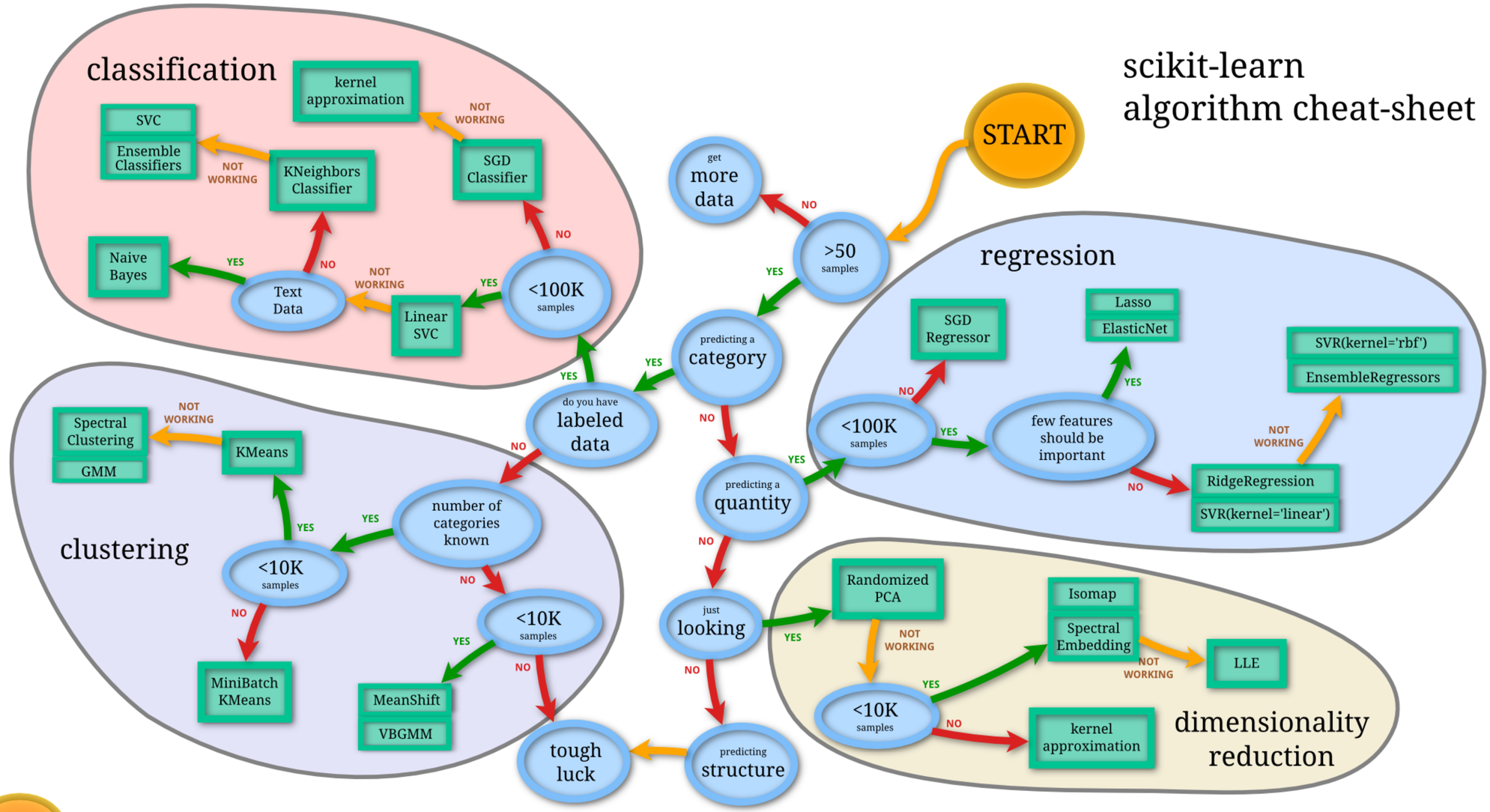
Machine Learning

Step 1

Step 2

Step 3

Step 4



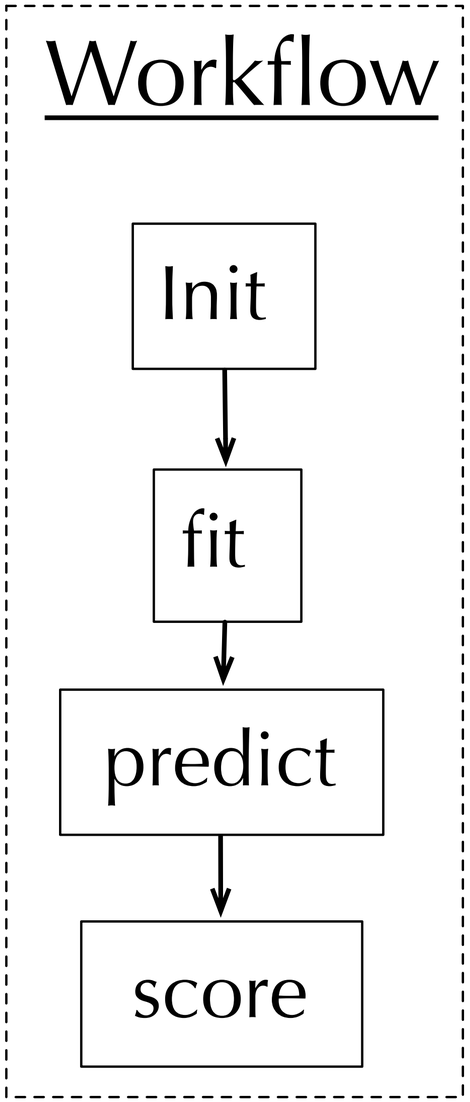
Machine Learning

Step 1

Step 2

Step 3

Step 4

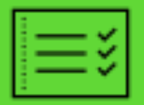



```python
# Import datasets, classifiers and performance metrics
from sklearn import datasets, svm, metrics
# The digits dataset
digits = datasets.load_digits()
# Create a classifier: a support vector classifier
classifier = svm.SVC(gamma=0.001)
# We learn the digits on the first half of the digits
classifier.fit(data[:n_samples // 2], digits.target[:n_samples // 2])
# Now predict the value of the digit on the second half:
expected = digits.target[n_samples // 2:]
predicted = classifier.predict(data[n_samples // 2:])
print("Classification report for classifier %s:\n%s\n"
      % (classifier, metrics.classification_report(expected, predicted)))
print("Confusion matrix:\n%s" % metrics.confusion_matrix(expected, predicted))
```

Computational Graph


Step 1

Step 2


Step 3

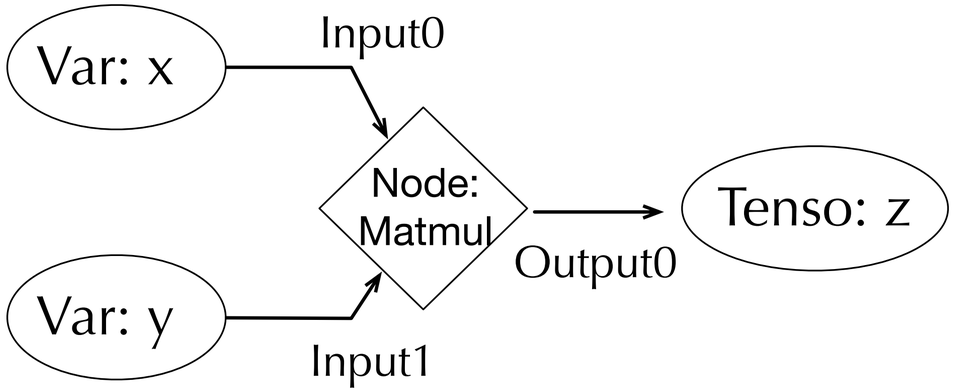

Step 4

Deep learning is matrix manipulation

Tensorflow, Theano

```
import tensorflow as tf

x = tf.placeholder(dtype='float32', shape=(2, 3))
y = tf.placeholder(dtype='float32', shape=(3, 4))
z = tf.matmul(x, y)
print(z)
```

```
Tensor("MatMul:0", shape=(2, 4), dtype=float32)
```



Numpy, PyTorch

```
import numpy as np

x = np.random.rand(2, 3)
y = np.random.rand(3, 4)
z = np.dot(x, y)
print(z)
```

```
[[ 0.78686185  0.61629636  0.59583339  1.04375596]
 [ 1.48166385  1.08328796  1.10720437  1.58379081]]
```

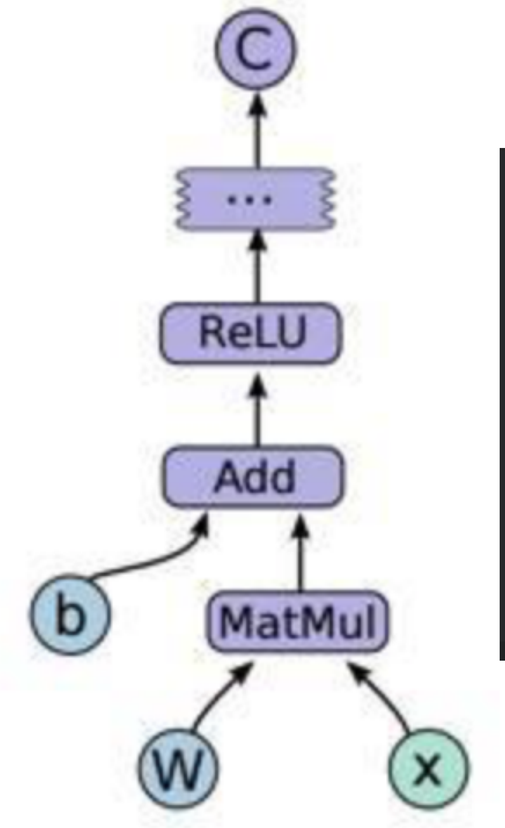
Computational Graph

Step 1

Step 2

Step 3

Step 4



Computational graph of
a feed forward layer

```tensorflow
import tensorflow as tf

# define two matrices
matrix1 = tf.constant([[3., 3.]])
matrix2 = tf.constant([[2.], [2.]])
# do multiplication
product = tf.matmul(matrix1, matrix2)
# Let's get a new session
sess = tf.Session()
result = sess.run(product)
print(result)
# the output is : [[ 12.]]
```

Tensorflow code

```keras
from keras.models import Sequential
from keras.layers import Dense, Activation
model = Sequential()
model.add(Dense(2, input_dim=3))
model.add(Activation('relu'))
```

Keras code

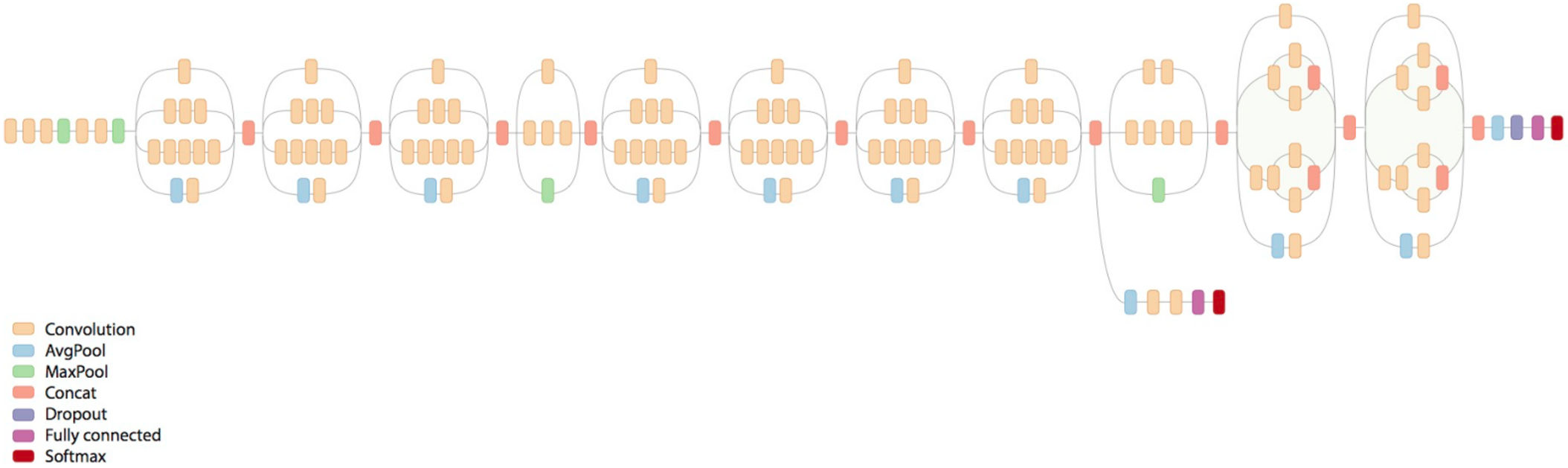
Computational Graph

Step 1

Step 2

Step 3

Step 4




```
from tensorflow.contrib import slim
logits, endpoints = slim.inception.inception_v3(
    images,
    dropout_keep_prob=0.8,
    num_classes=num_classes,
    is_training=for_training,
    restore_logits=restore_logits,
    scope="Inception")
```

Evaluation

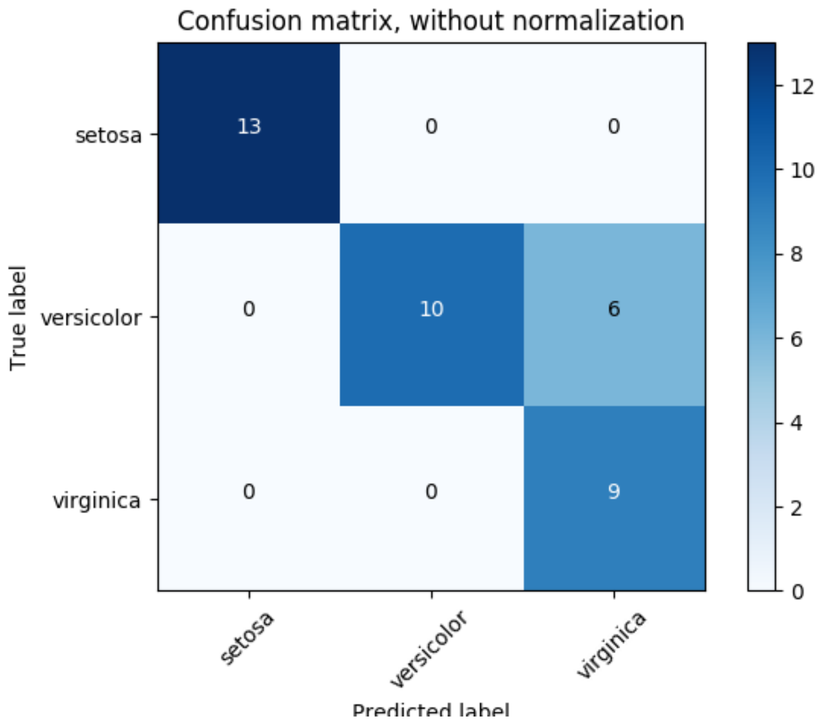
Step 1

Step 2

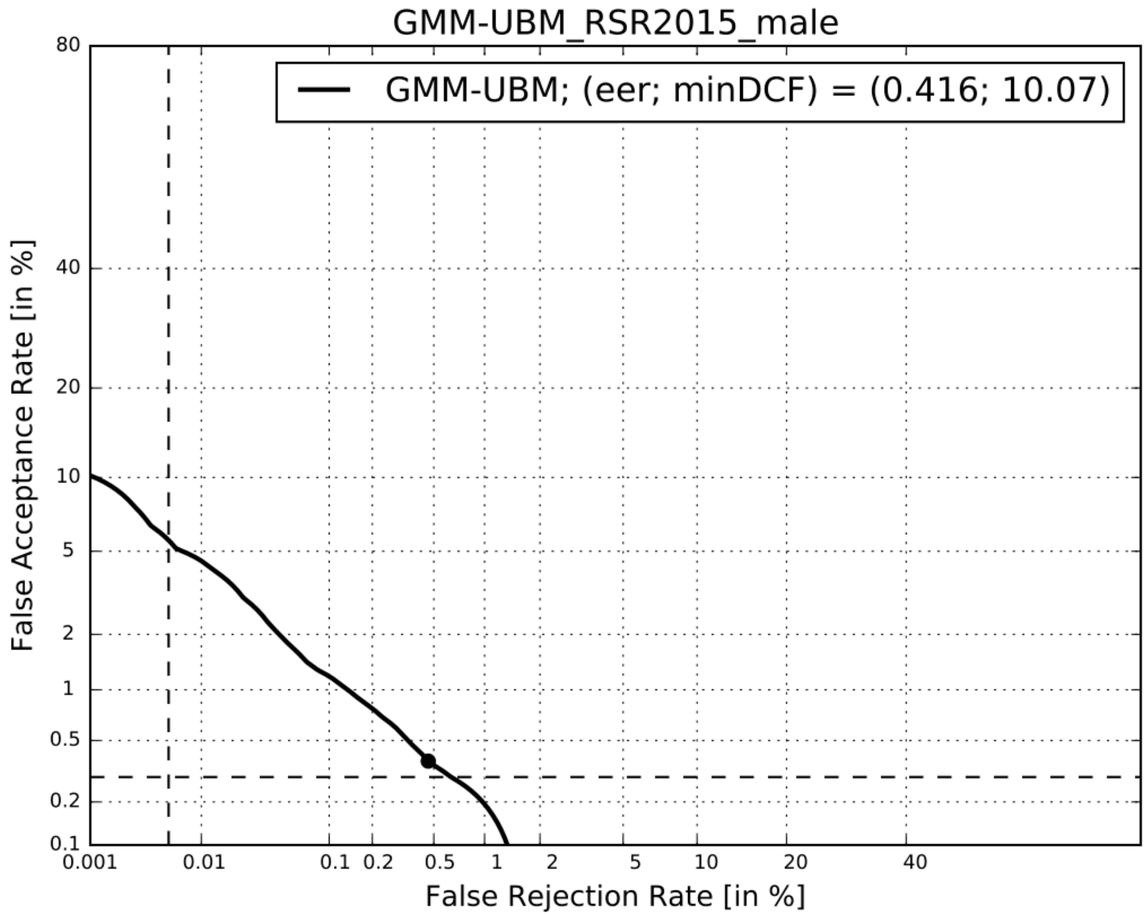
Step 3

Step 4

Confusion matrix (sklearn)



EER (implemeted in sidekit)



Git: version control

Step 1

Step 2

Step 3

Step 4