Postgraduaat

AI Technical Architect

academiejaar ‘23-’24

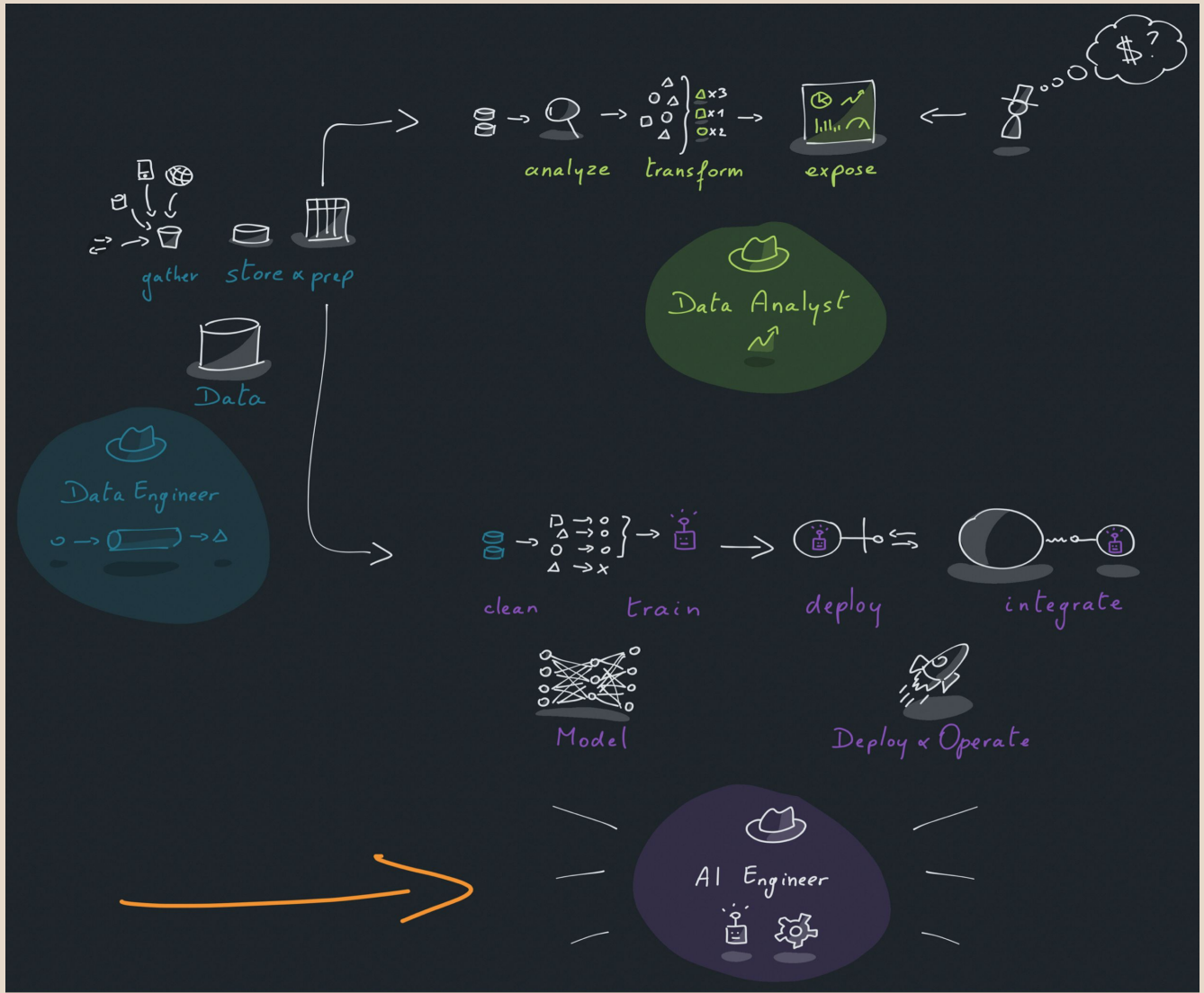
Postgraduaat

AI Technical Architect

Technologies
Hands on day 1

Tim Dupont
Sam Van Rijn

08 jan ‘24



PXL NEXT

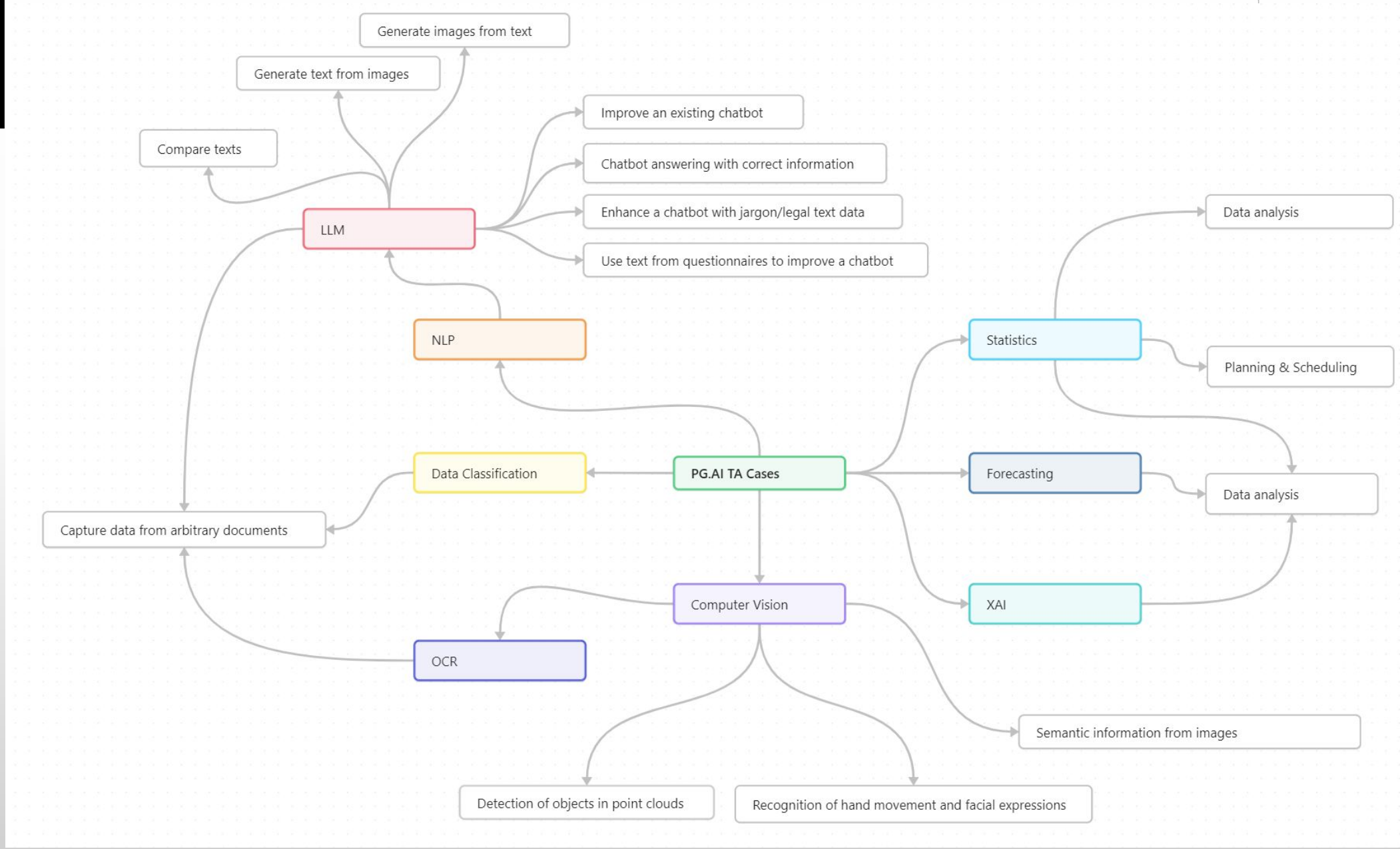
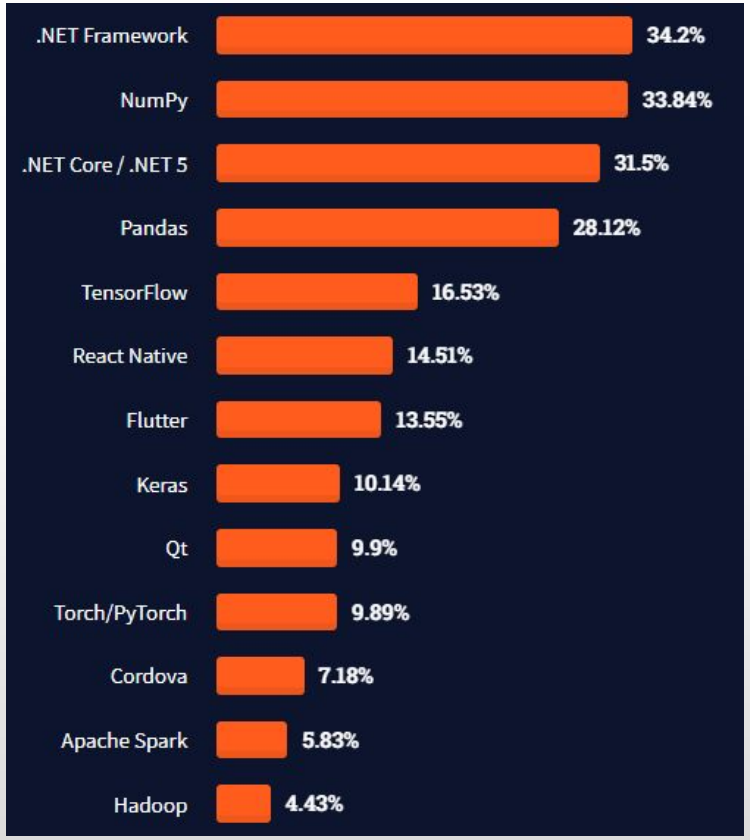

Generate images from text
Generate text from images
Compare texts
LLM
Improve an existing chatbot
Chatbot answering with correct information
Enhance a chatbot with jargon/legal text data
Use text from questionnaires to improve a chatbot
NLP
Data Classification
Capture data from arbitrary documents
PG.AI TA Cases
Statistics
Data analysis
Planning & Scheduling
Forecasting
Data analysis
XAI
Computer Vision
OCR
Semantic information from images
Detection of objects in point clouds
Recognition of hand movement and facial expressions

Table of contents

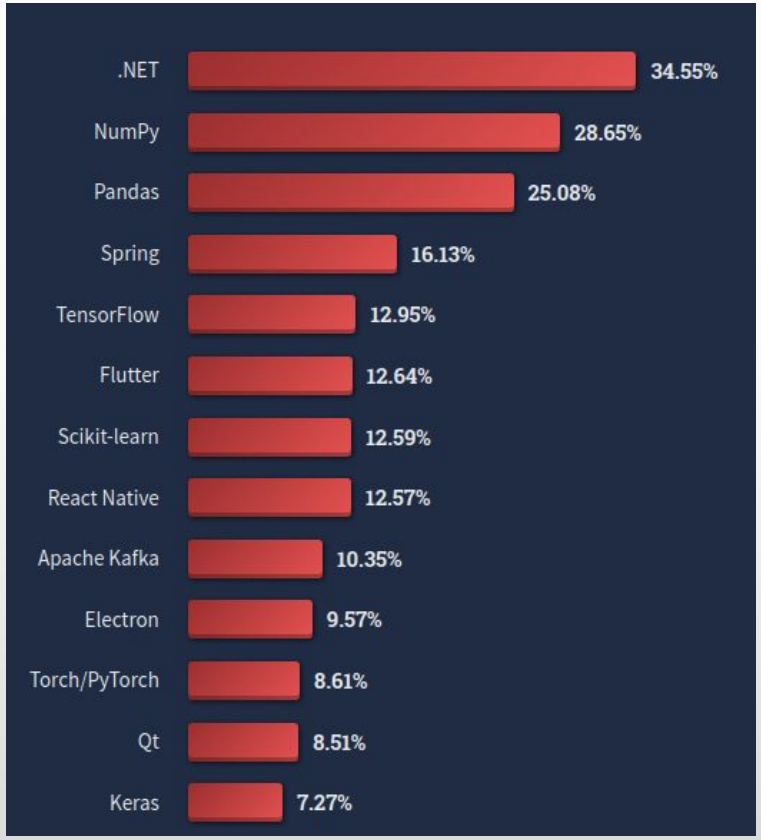
Frameworks & libraries

Most popular frameworks, libraries & tools

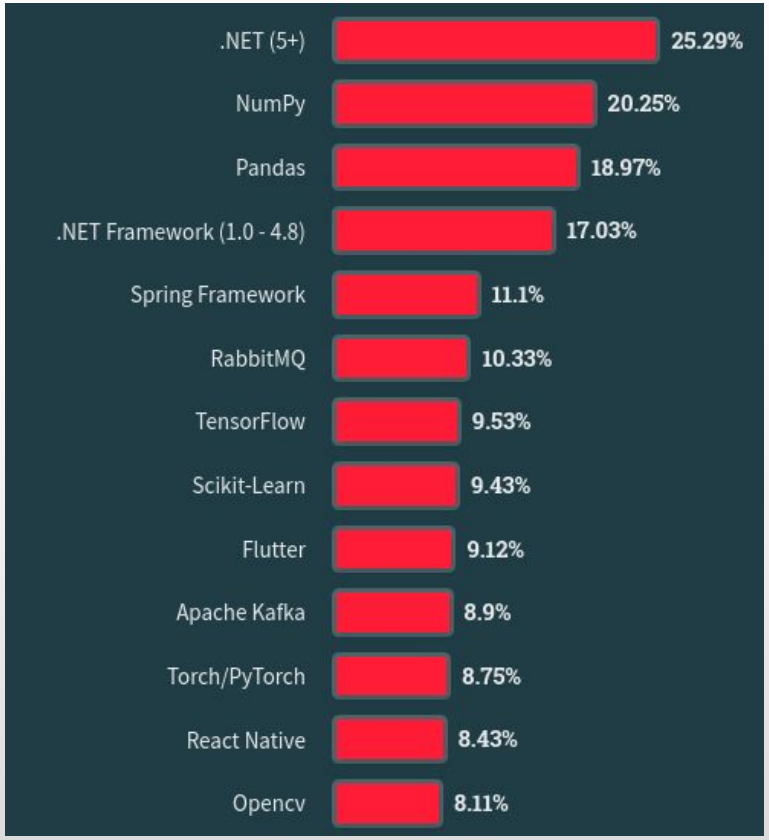
2021



2022



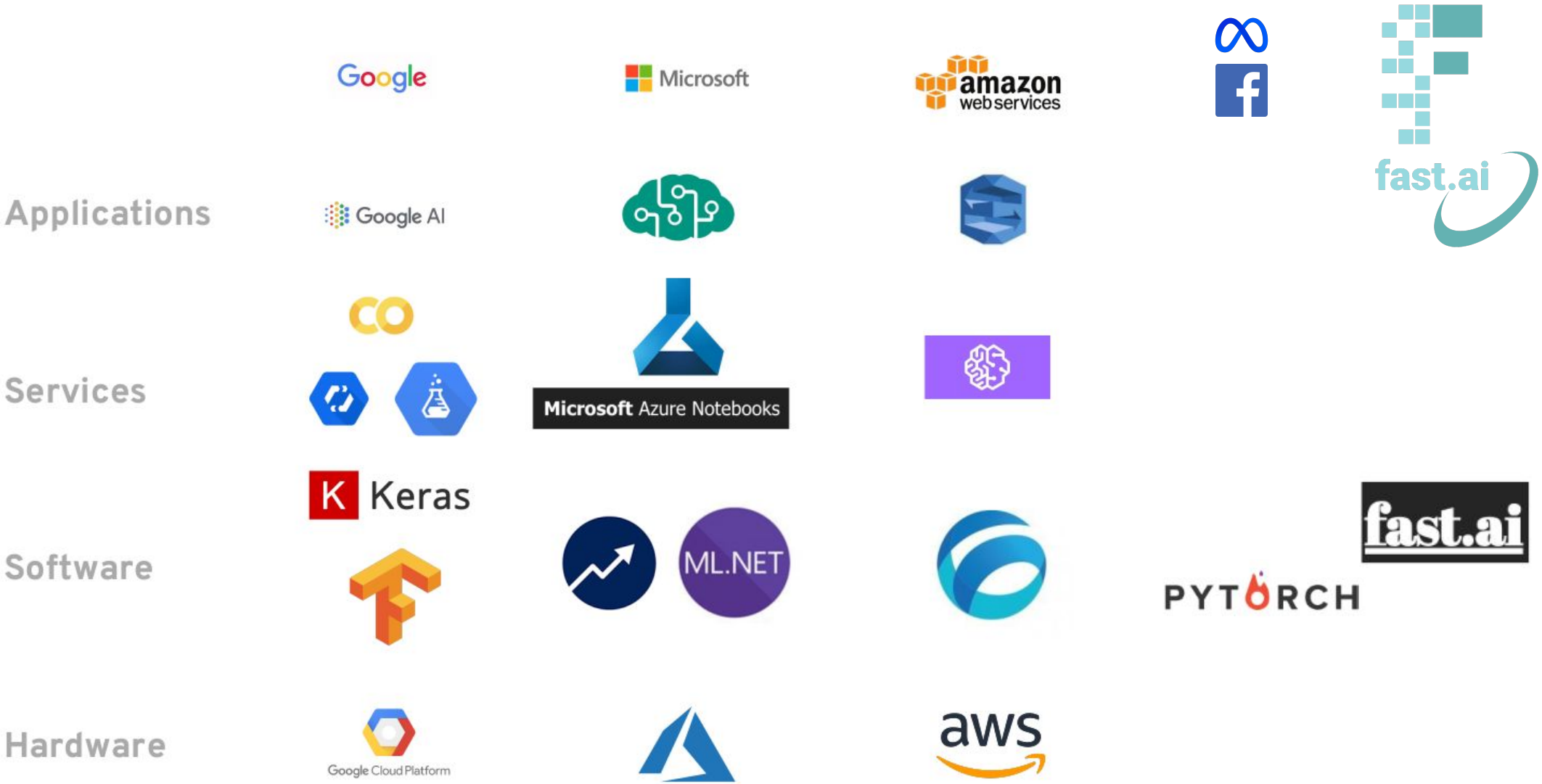
2023



[INFO]

Source: https://insights.stackoverflow.com/survey/2021 & https://survey.stackoverflow.co/2022 & https://survey.stackoverflow.co/2023

AI software stack



Python - Usage

Web development

django

django CMS

FastAPI

Education

Scientific and Numeric

polars

matplotlib

pandas

PyTorch

TensorFlow

fast.ai

Business Applications

Tryton

odoo

Prototyping

(Smart) Robotics

Cloud & Automation

PXL NEXT

Frameworks & libraries

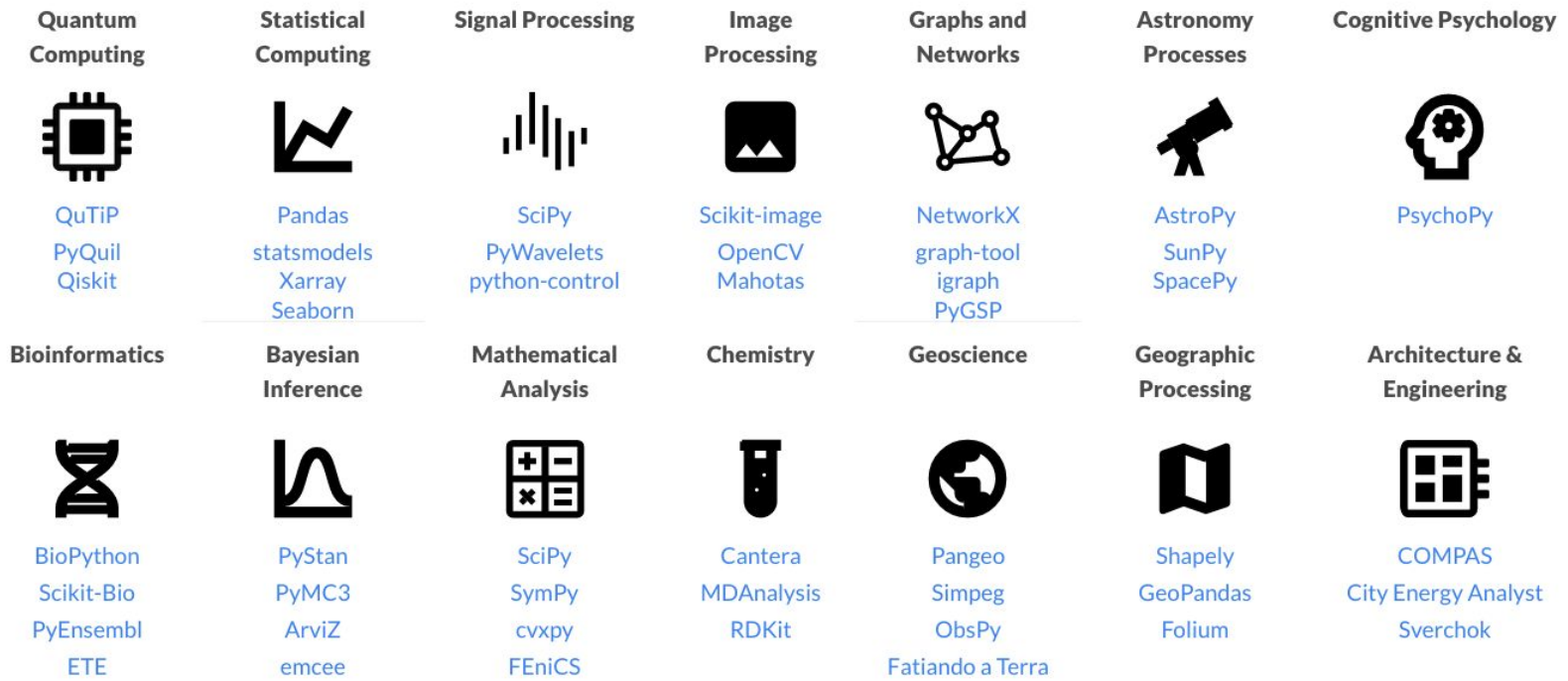
for Data Exploration & Model Training (classical Machine Learning)

Data & ML

Not Python

NumPy

- The fundamental package for scientific computing with Python
- Written in C and Python
  (In other words: “behind the scenes” is optimized, pre-compiled C code)
- Multidimensional array object
- Assortment of routines for fast operations on arrays
  (incl. manipulations, sorting, I/O, linear algebra, statistical operations, …)
- Extensively used



[INFO]
More info: https://numpy.org

pandas

- Data analysis and manipulation tool (library)
- Written in Python, C (and of course it uses NumPy)
- Fast, powerful, flexible and easy to use
  - DataFrame object for data manipulation
  - Reading and writing data between in-memory data structures and different file formats.
  - Data alignment and integrated handling of missing data.
  - Reshaping and pivoting of data sets.
  - Label-based slicing, fancy indexing, and sub-setting of large data sets.
  - Column insertion & deletion and data set merging & joining
  - Time series-functionality

```
df = df.dropna()
```

```
df['Age'] = df['Age'].fillna(df['Age'].median())
```

```
df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])
```

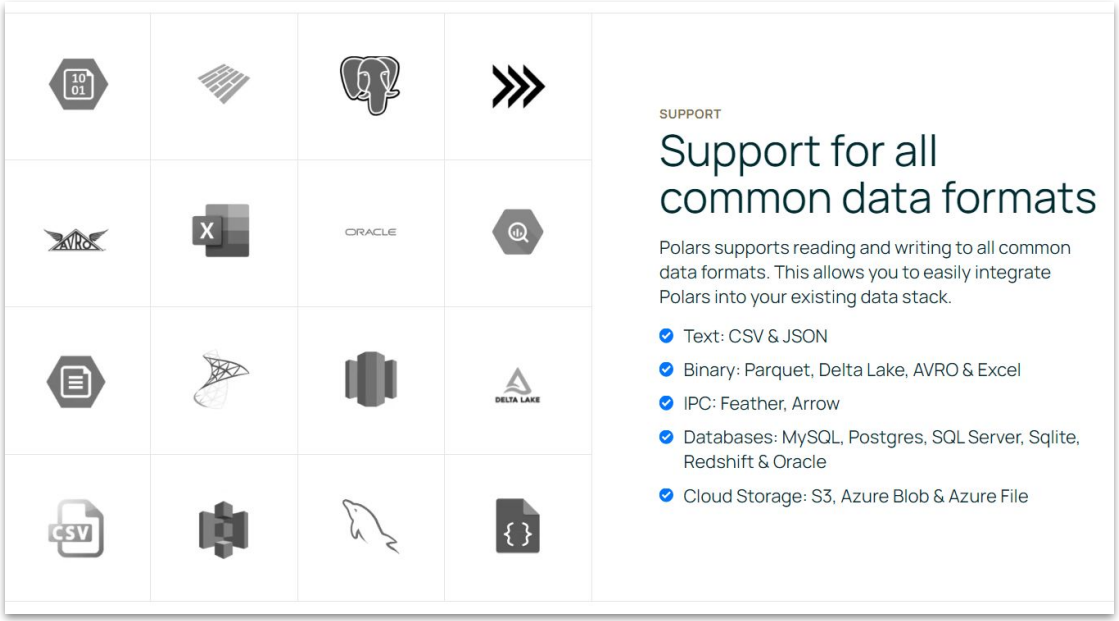
```
for column in obj_columns:
    df[column] = df[column].astype('category')
```

```
df = pd.get_dummies(df, columns=['Embarked', 'Title'])
```

[INFO]
More info: https://pandas.pydata.org

Polars

- Blazingly fast DataFrames in Rust, Python, Node.js, R and SQL
- Written in Rust
- Blazingly fast, lightweight…
  - Lazy | eager execution
  - Multi-threaded
  - SIMD
  - Query optimization
  - Powerful expression API
  - Hybrid Streaming (larger than RAM datasets)
  - Rust | Python | NodeJS | R | …




```
import polars as pl

df = pl.read_csv("wowah_data.csv", parse_dates=False)

df.head(5)
```

[INFO]
More info: https://pola.rs

SciPy

- Python library used for scientific computing and technical computing
- Written in Python, Fortran, C, C++ (and of course it uses NumPy)
- Cross platform
- Modules for
  - Optimization
  - Linear algebra
  - Integration
  - Interpolation
  - Fast Fourier Transform (FFT)
  - Signal and image processing
  - Clustering
  - …

```
import scipy
from scipy.cluster import hierarchy as hc
corr = np.round(scipy.stats.spearmanr(X_keep).correlation, 4)
corr_condensed = hc.distance.squareform(1-abs(corr))
z = hc.linkage(corr_condensed, method='average')
fig = plt.figure(figsize=(16,10))
dendrogram = hc.dendrogram(z, labels=X_keep.columns,
                           orientation='left', leaf_font_size=16)
plt.show()
```

[INFO]
More info: https://scipy.org and https://github.com/scipy/scipy

scikit-learn

- Written in Python, C and C++ (and of course it uses NumPy)
- Supports most of the classical supervised and unsupervised learning algorithms:
  - Random forests
  - SVM
  - Naive Bayes
  - Gradient boosting
  - Clustering
- Designed to interoperate with the numerical and scientific libraries NumPy and SciPy
- Shipped with Anaconda

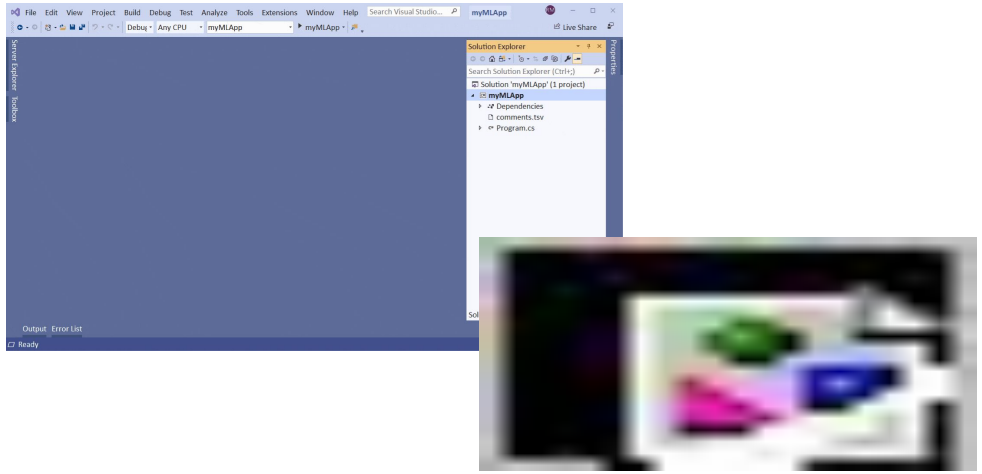
Advantages	Disadvantages
Great classical machine learning support	Very limited support for Neural Networks
Easy to use and excellent for beginners	No Deep Learning support

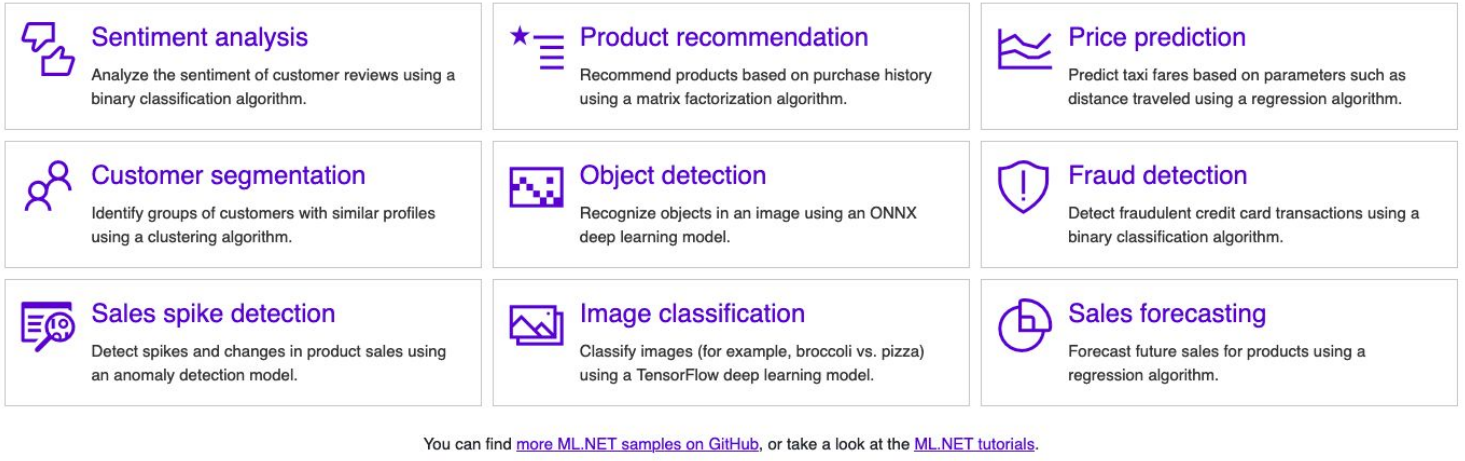
[INFO]
More info: https://scikit-learn.org/stable/ and https://github.com/scikit-learn/scikit-learn

ML.NET

- Model-based Machine Learning analytic and prediction capabilities
- For existing .NET developers (C# and F#)
- Written in C# and C++
- Built upon .NET Core and .NET Standard
- Cross platform (Linux, macOS, and Windows)
- Model Builder (a simple UI tool) and ML.NET CLI
- Documentation used to be awful, but it's getting better

  (Intro: https://learn.microsoft.com/en-us/dotnet/machine-learning/how-does-mldotnet-work)
  (Samples: https://github.com/dotnet/machinelearning-samples)





[INFO]
More info: https://dotnet.microsoft.com/en-us/apps/machinelearning-ai/ml-dotnet and https://github.com/dotnet/machinelearning

PXL NEXT

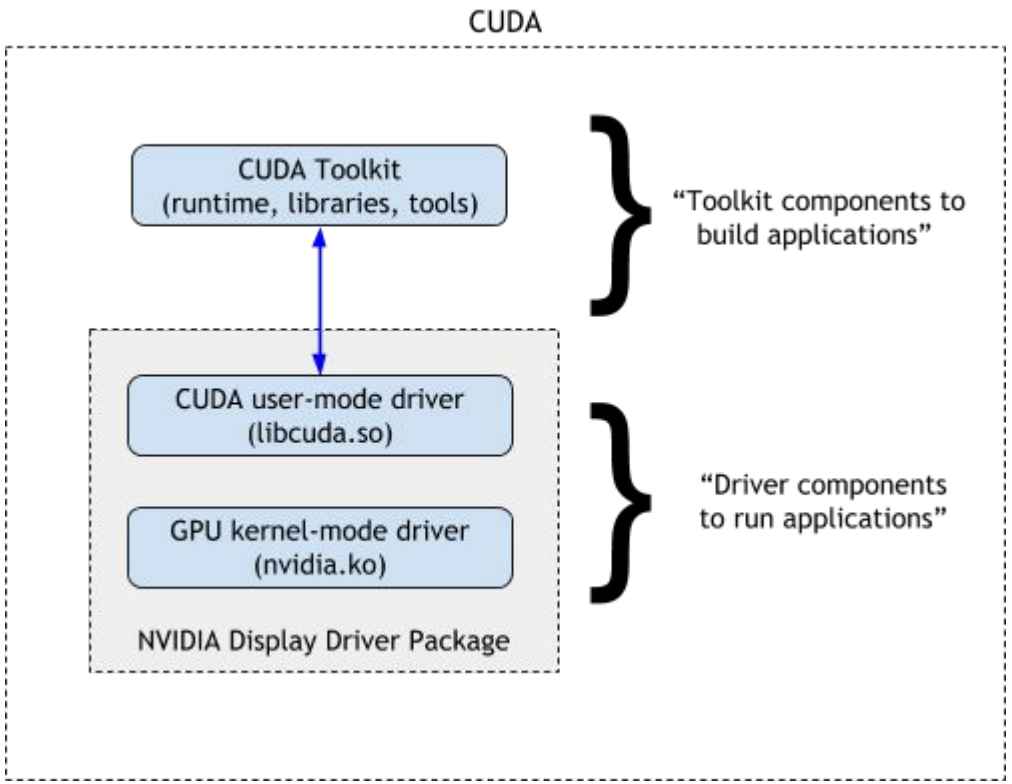
Frameworks & libraries

for Data Exploration & Model Training (Deep Learning)

Deep learning

Not strictly Python

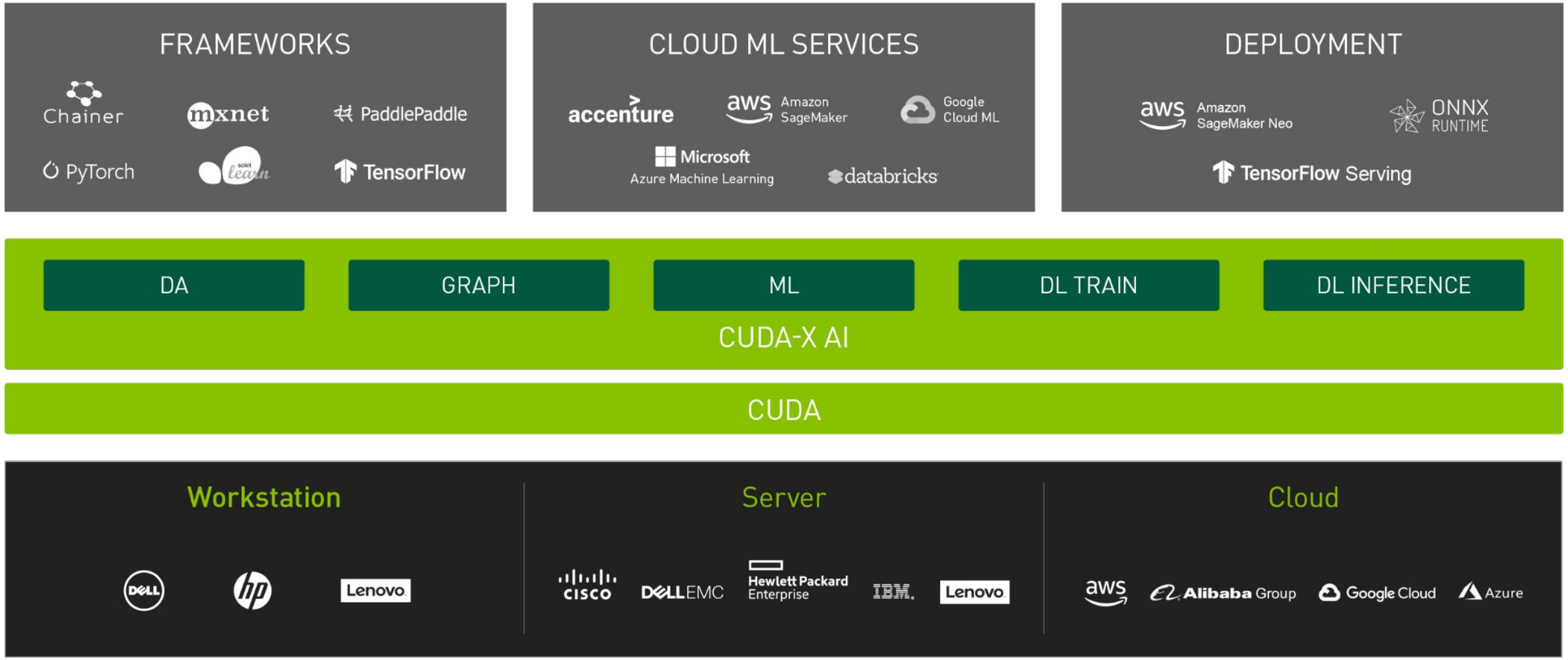
Nvidia CUDA



- CUDA (Compute Unified Device Architecture)
- Parallel computing platform
  + Application Programming Interface (API)
- Use GPU for general purpose processing
  (GPU has parallel processors called CUDA Cores
  example: GeForce RTX 3090 has 10496 cores)
- libcudnn: C Lib for Deep learning
  (Deep Neural Network library)
- **By far the most popular AI computing platform!**
  Supported by most AI libraries:
  Keras, MATLAB, PyTorch, TensorFlow, …
- **Only for Nvidia graphics cards!**
  (OpenCL is an open alternative, but less popular)

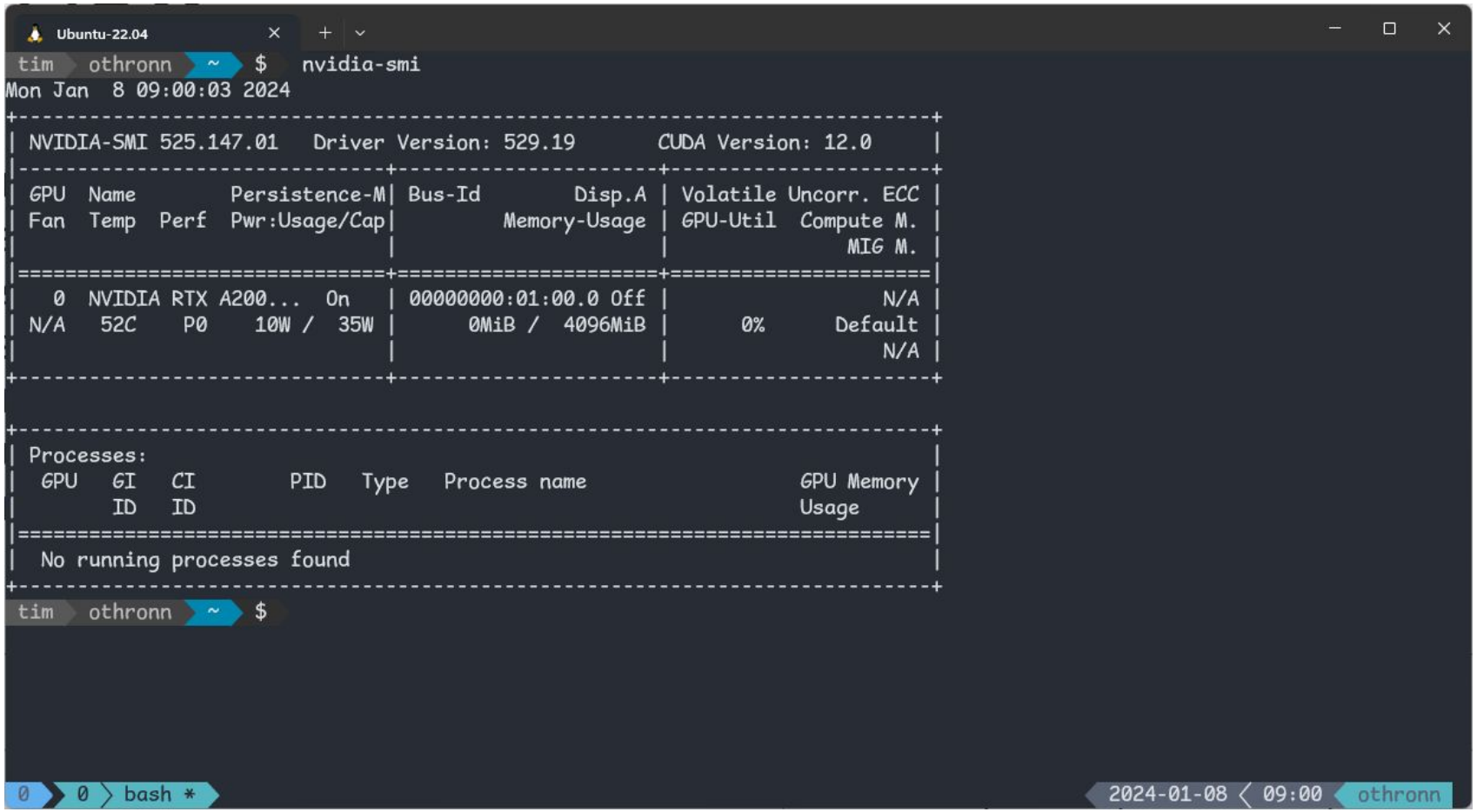
[INFO]
More info: https://developer.nvidia.com/cuda-toolkit

Nvidia CUDA



[INFO]
More info: https://www.nvidia.com/en-us/technologies/cuda-x

Nvidia CUDA



Checking GPU with `nvidia-msi` in WSL2 (Ubuntu-22.04 on Windows 11).

[INFO]
Check with: `nvidia-msi`

Nvidia CUDA

Checking GPU with `nvidia-msi` on Google Colab

[INFO]
Check with: `nvidia-msi`

TensorFlow

- Python
- Created by Google
- Written in Python, C++ (and of course it uses NumPy)
- Makes use of CUDA
- Tensor computation (like NumPy) with strong GPU acceleration
- Stateful dataflow graphs
- Tensor Processing Unit (TPU) → dedicated hardware support

Advantages	Disadvantages
Fast	Not so easy to use (too low level)
Neural networks and Deep Learning support	

[INFO]
More info: https://www.tensorflow.org and https://github.com/tensorflow/tensorflow

Keras

- Python library for artificial neural networks
- On top of TensorFlow a.k.a. interface for the TensorFlow library
  (Until version 2.3 also support for Microsoft Cognitive Toolkit, Theano, or PlaidML.)
- Strong ties with Google

Advantages	Disadvantages
Machine Learning and Deep Learning support	Not enough documentation
Easy to use	

```
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))

# The final layer has 10 nodes, each node represents one class of numbers
network.add(layers.Dense(10, activation='softmax'))
```

```
network = models.Sequential()
network.add(layers.Dense(300, activation='relu', input_shape=(28 * 28,), kernel_regularizer=keras.regularizers.l1_l2(l1=0, l2=0.001)))
#network.add(layers.Dropout(0.4))
network.add(layers.Dense(100, activation='relu'))#, kernel_regularizer=l2(0.1)))
#network.add(layers.Dense(300, activation='relu', kernel_regularizer=l2(0.0001)))
#network.add(layers.Dropout(0.2))

# The final layer has 10 nodes, each node represents one class of numbers
network.add(layers.Dense(10, activation='softmax'))
```

[INFO]
More info: https://keras.io and https://github.com/keras-team/keras

PyTorch

- Written in Python, C++, CUDA (and of course it uses NumPy)
- Created by Facebook
- Tensor computation (like NumPy) with strong GPU acceleration
- Deep neural networks built on a tape-based autodiff system

Advantages	Disadvantages
Machine Learning and Deep Learning support	Not enough documentation about some methods and parameters
Allows customization	

[INFO]
More info: https://pytorch.org and https://github.com/pytorch/pytorch

fast.ai

- Written in Python (and of course it uses NumPy)
- Built on top of Scikit-learn and PyTorch
- Aim is to make Machine and Deep Learning as accessible as possible
- Contains a lot of functions to make data processing easier

Advantages	Disadvantages
Great all round support	API changes quite often
Integrates new techniques and methods rapidly	Less market usage
Great for learning concepts easily	

[INFO]
More info: https://www.fast.ai and https://github.com/fastai/fastai

Frameworks & libraries

for Visualizations

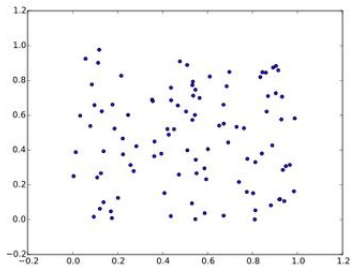
Visualizations

matplotlib

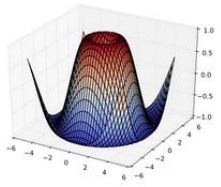
Not Python

Matplotlib

- Python library for creating static, animated, and interactive visualizations
- Written in Python
- Makes extensive use of NumPy
- Embeddable in GUI toolkits (i.e. QT)



Scatter plot



3D plot

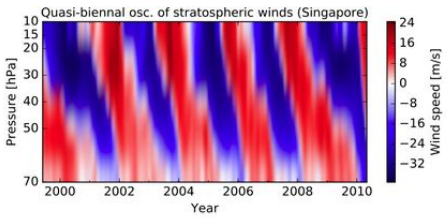
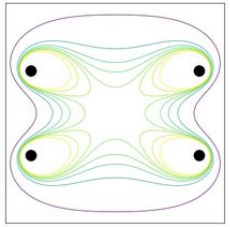
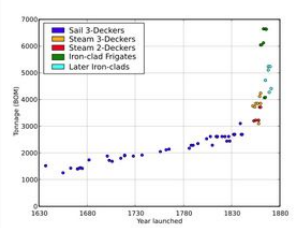


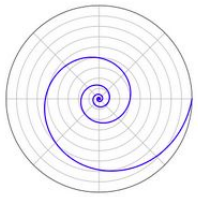
Image plot



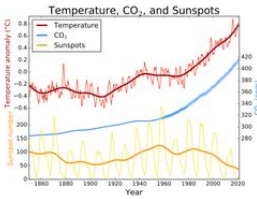
Contour plot



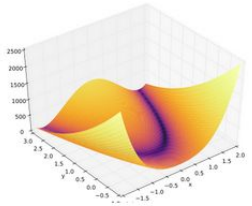
Scatter plot



Polar plot



Line plot



3-D plot

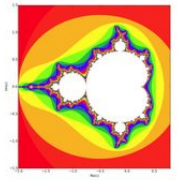
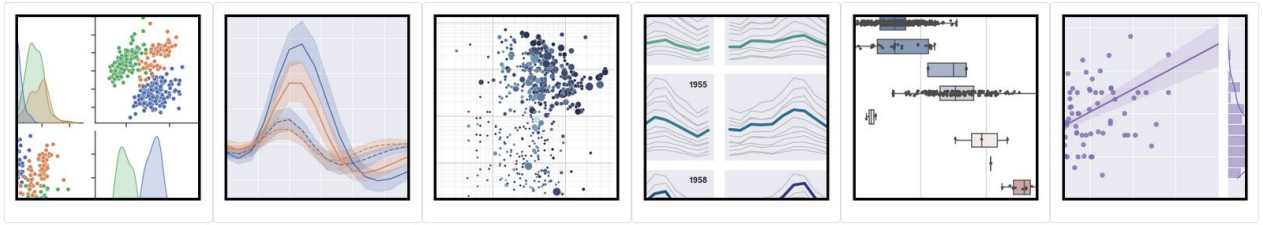


Image plot

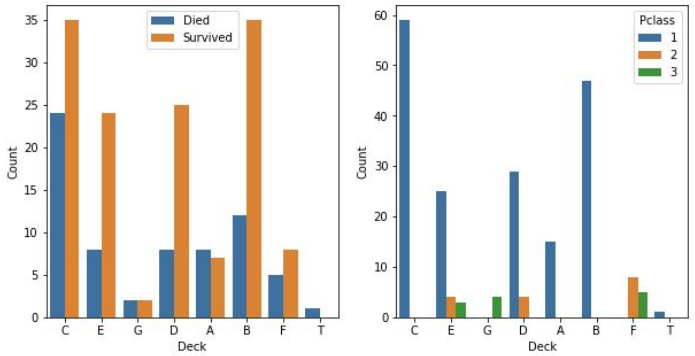
[INFO]
More info: https://matplotlib.org

Seaborn



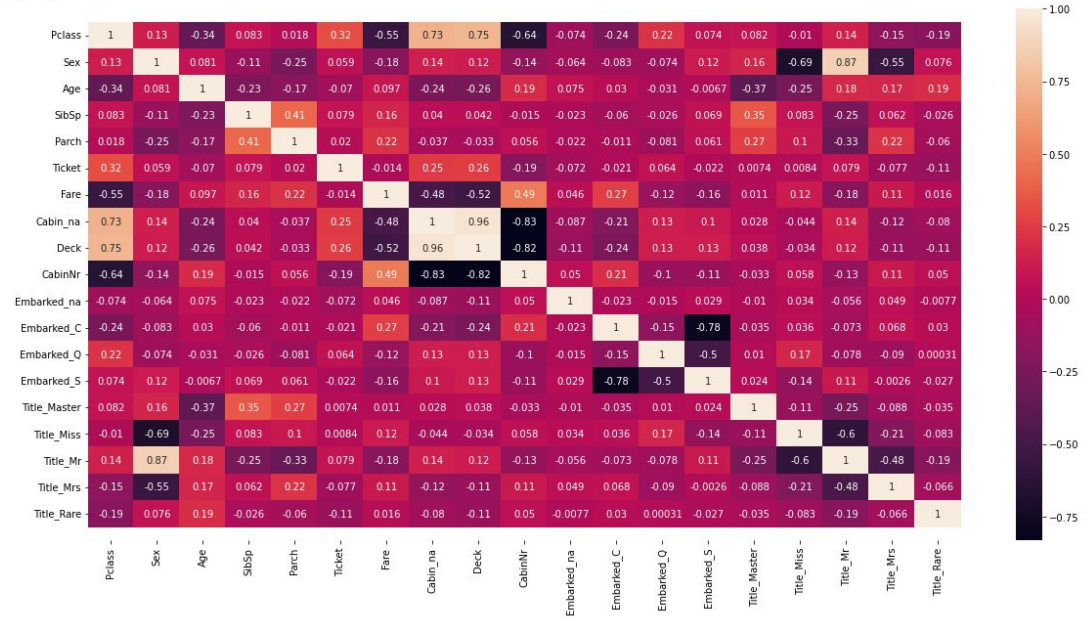
- Data visualization library based on Matplotlib
- High-level interface for drawing attractive and informative statistical graphics
- Integrates closely with pandas data structures

```
fig, ax = plt.subplots(1,2, figsize=(10,5))

sns.countplot('Deck', data=df[df['Deck'] != 'Z'], hue='Survived', ax=ax[0])
sns.countplot('Deck', data=df[df['Deck'] != 'Z'], hue='Pclass', ax=ax[1])

ax[0].set_xlabel('Deck')
ax[0].set_ylabel('Count')
ax[0].legend(['Died', 'Survived'])

ax[1].set_xlabel('Deck')
ax[1].set_ylabel('Count')

fig.show()
```




```
fig, ax = plt.subplots(figsize = (20, 10))
# we drop the feature 'Survived', since this is our dependent variable (the variable we are trying to predict)
df_corr = df.drop(['Survived'], axis=1).corr()
fig = sns.heatmap(df_corr, annot = True)

# This is a fix for a bug in the visualization library
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

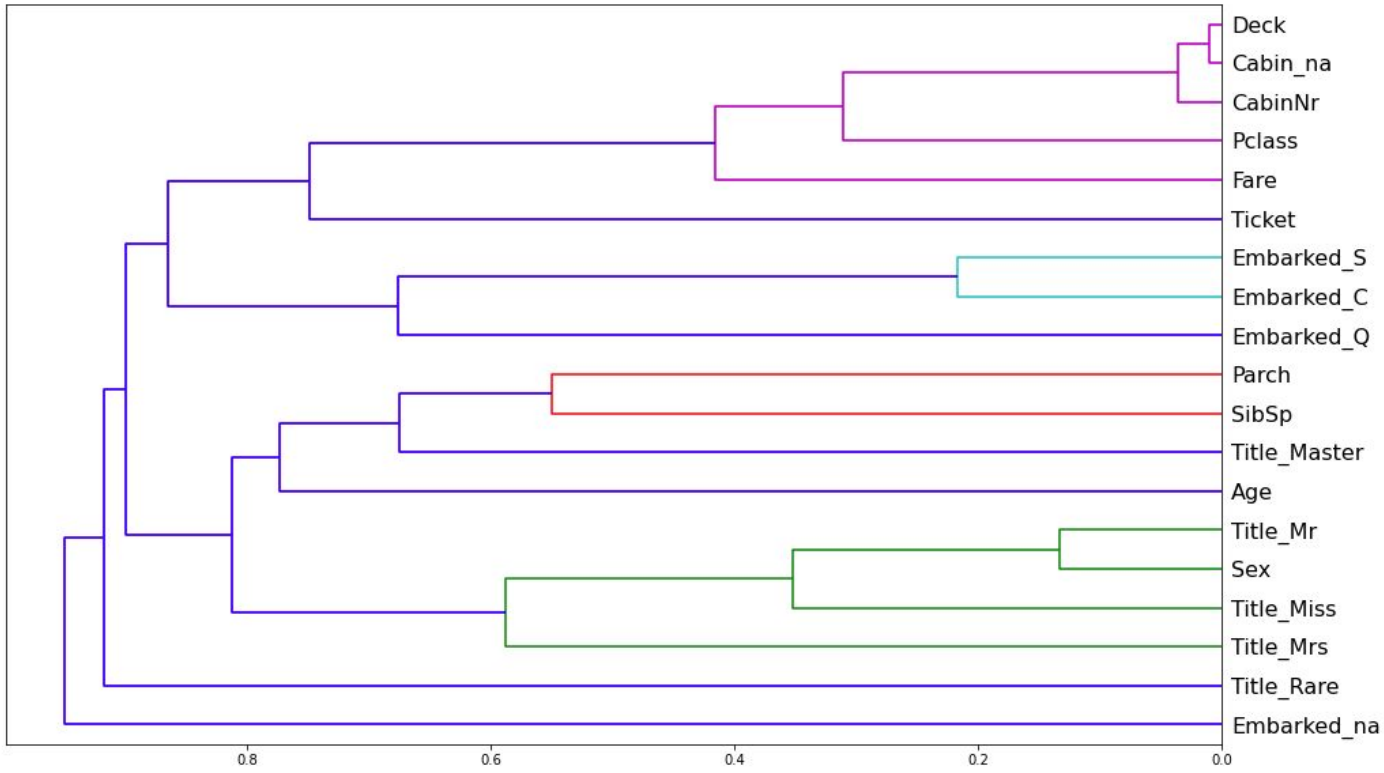
```
(19.5, -0.5)
```



[INFO]
More info: https://seaborn.pydata.org/introduction.html

SciPy


```
import scipy
from scipy.cluster import hierarchy as hc
corr = np.round(scipy.stats.spearmanr(df.drop(['Survived'], axis=1)).correlation, 4)
corr_condensed = hc.distance.squareform(1-abs(corr))        # we use 1-abs(corr) instead of 1-corr, to incorporate negative correlations as well as positive ones
z = hc.linkage(corr_condensed, method='average')
fig = plt.figure(figsize=(16,10))
dendrogram = hc.dendrogram(z, labels=df.columns, orientation='left', leaf_font_size=16)
plt.show()
```



[INFO]

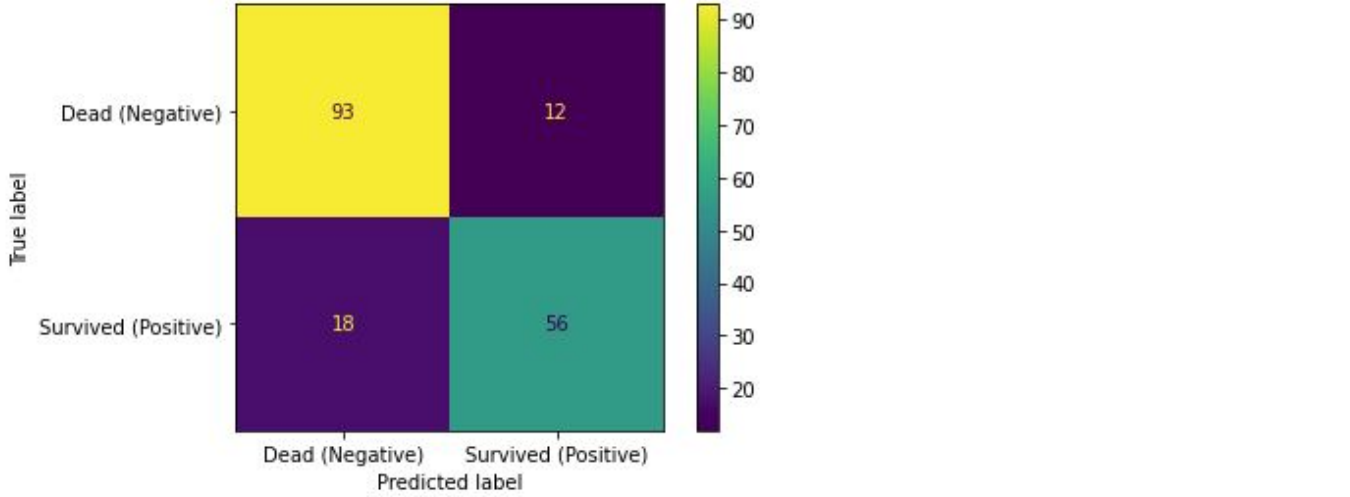
More info: https://scipy.org and https://github.com/scipy/scipy

scikit-learn


```
RF_preds = RF_model.predict(X_test)
RF_preds_rounded = np.round_(RF_preds, 0)
RF_cm = confusion_matrix(y_test, RF_preds_rounded)
```

```
disp = ConfusionMatrixDisplay(confusion_matrix=RF_cm,
                              display_labels=['Dead (Negative)',
                                              'Survived (Positive)'])
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fe34696f310>
```



[INFO]
More info: https://scikit-learn.org/stable/ and https://github.com/scikit-learn/scikit-learn

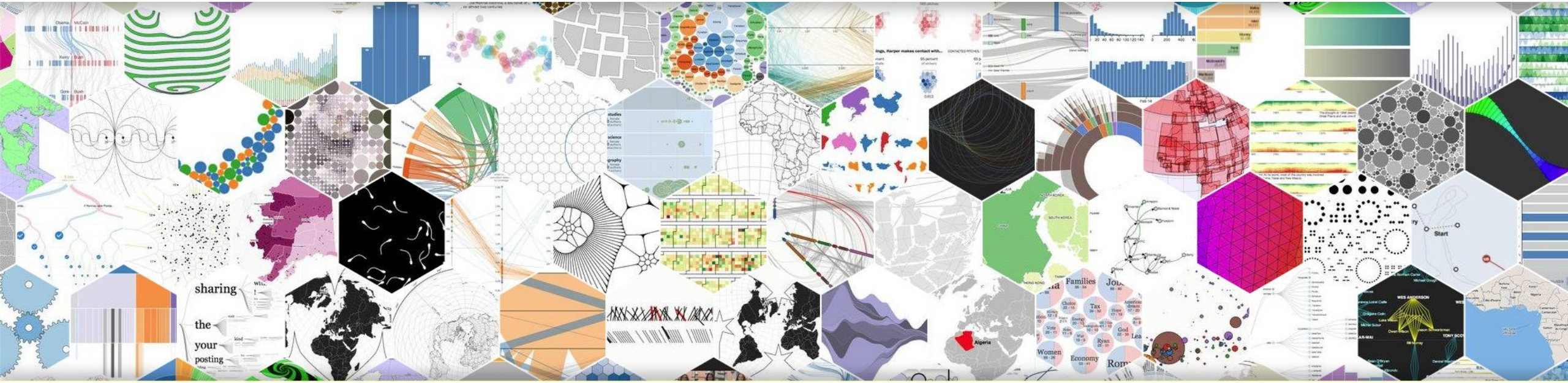
plotly

- Interactive, open-source, and browser-based graphing library for Python
- Built on top of plotly.js,
- Declarative charting library.
- Over 30 chart types, including:
  - Scientific charts
  - 3D graphs
  - Statistical charts
  - SVG maps
  - Financial charts
  - …

[INFO]
More info: https://plotly.com/python

Data-Driven Documents (d3.js)

- JavaScript library for manipulating documents based on data.
- Using HTML, SVG, and CSS.
- Emphasis on web standards



[INFO]
More info: https://d3js.org

PXL NEXT

Frameworks & libraries

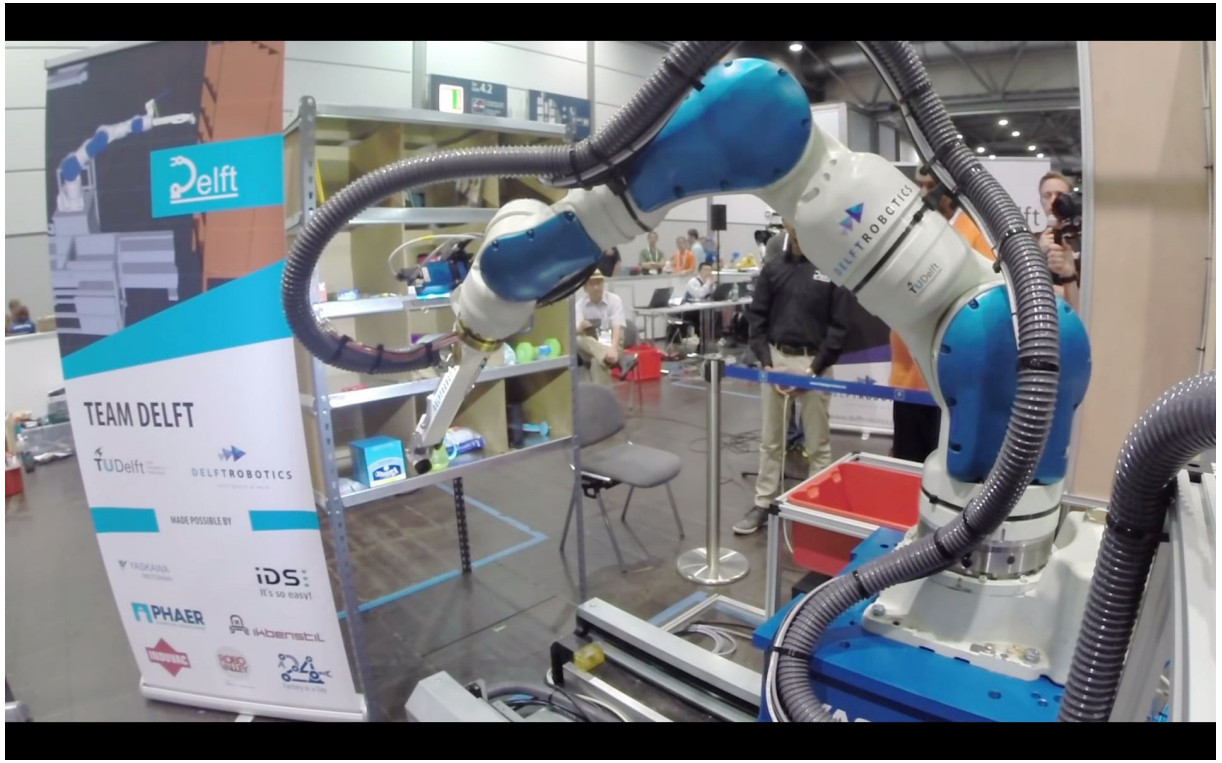
Prototyping & Robotics

Prototyping & Robotics

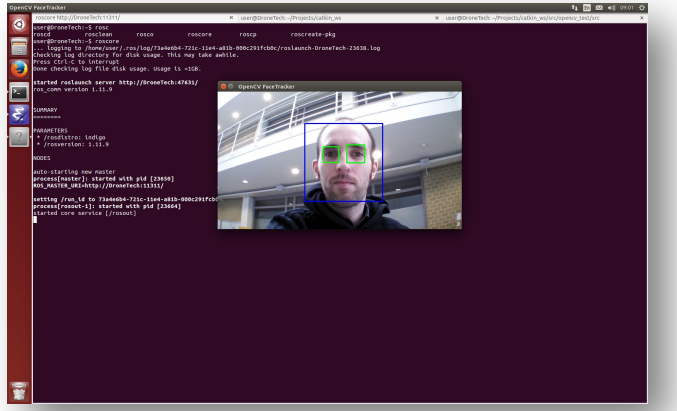
Not strictly Python

PyQt

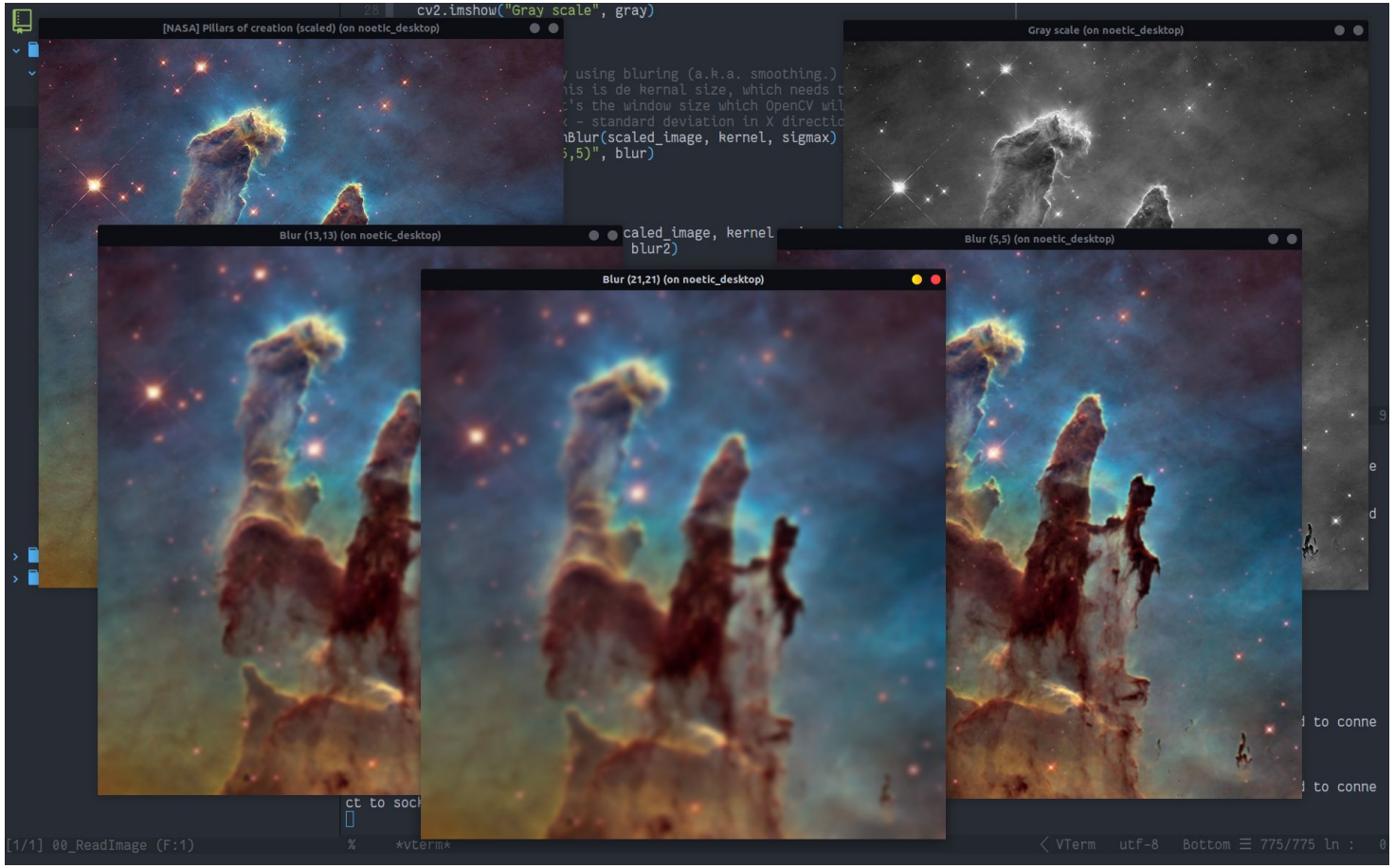
ROS



OpenCV



OpenCV



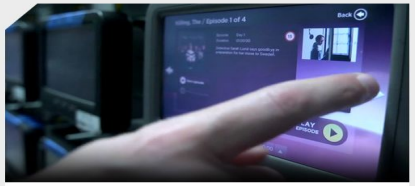
- Open Source Computer Vision
- Cross-platform library
- Real-time Computer Vision
- Originally by Intel
- Well documented
- Multiple bindings
  (C++, Java & Python)

[INFO]
More info: https://opencv.org

OpenCV in 4h

[INFO]
Source: https://www.youtube.com/watch?v=oXlwWbU8l2o

QT



Panasonic Avionics – Inflight Infotainment

Read More

Veriskin – Skin Cancer Diagnostics Device

Read More

Rimac Automobili – Car Infotainment System

Read More

AutoIO – Digital Instrument Clusters

Read More

BEP Marine – Marine Automation Systems

Read More

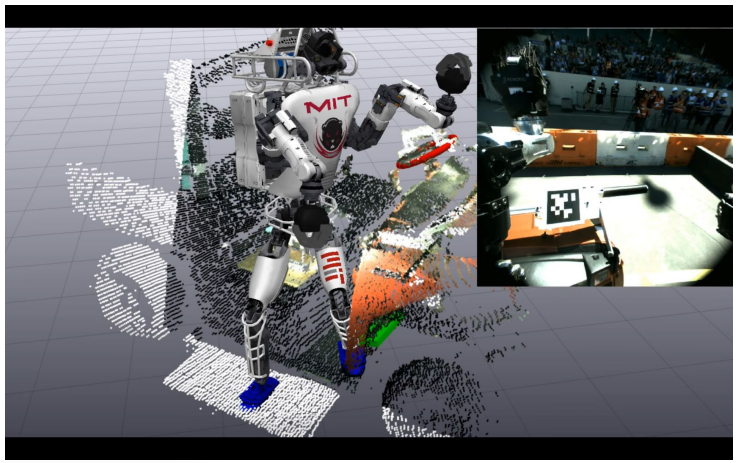
AMD – Graphics software

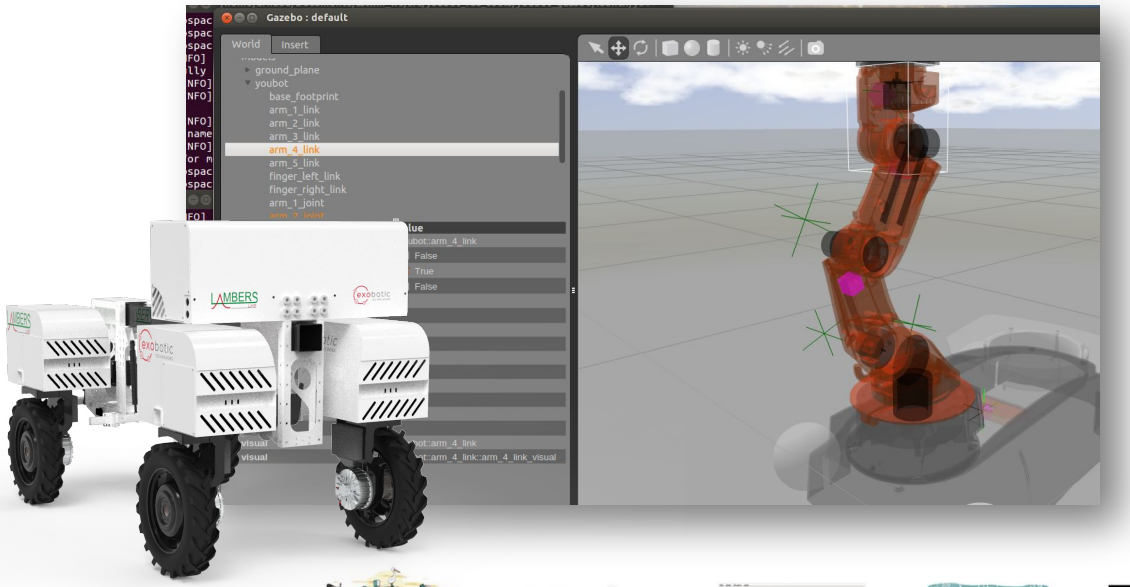
Read More

(pronounced "**cute**") is a **free and open-source widget toolkit** for creating graphical user interfaces as well as cross-platform applications that **run on various software and hardware platforms** such as Linux, Windows, macOS, Android or embedded systems with **little or no change in the underlying codebase** while still being a **native application with native capabilities and speed**.

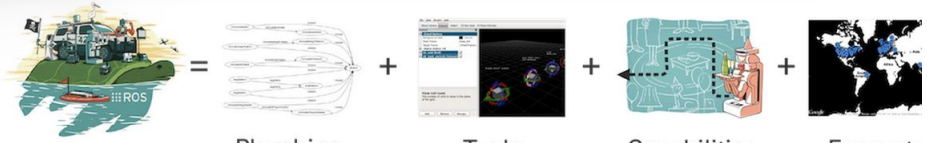
[**INFO**]
More info: https://www.qt.io

ROS





- Open source software
  (BSD licence)
- Robotics framework
- C++, Python, …
- “Flexible Robotics”
- Message framework
- Active Community
- Used in industry
  (BMW, Bosch, Siemens, Boeing, John Deere, CAT, Panasonic, …)



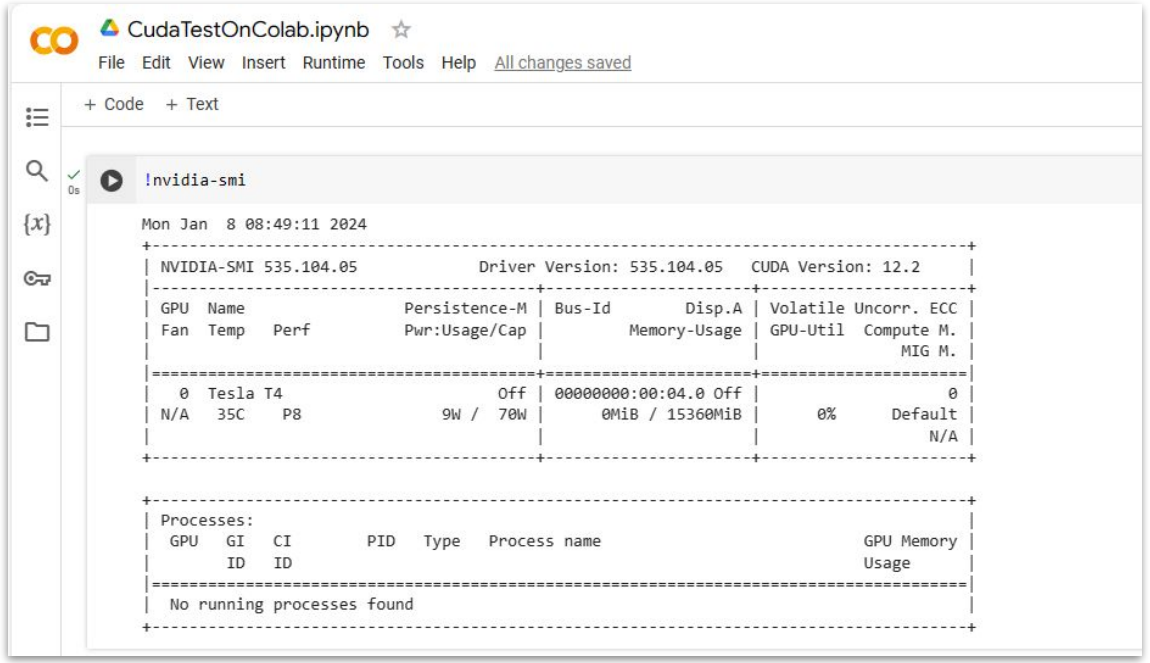
[INFO]
More info: https://www.ros.org

Hands-on exercises

Tools

Google Colaboratory

- Write and execute Python in your browser
- Jupyter Notebook like
- Zero configuration required
- Access to AI-accelerators free of charge
  - TPU
  - Nvidia T4 GPU
- Pay for extra resources
  - Nvidia V100 GPU
  - Nvidia A100 GPU
  - …
- Easy sharing
- Free:
  - Shared resources
  - Sessions are limited (time & resources)



[INFO]
More info: https://colab.research.google.com

Libraries used in the hands-on exercises

python™

pandas

NumPy

YOLO

seaborn

matplotlib

Keras

scikit learn

SciPy

OpenCV

Hands-on exercises

https://github.com/PXLAIRobotics/PG.AI-TA-2324-HandsOn