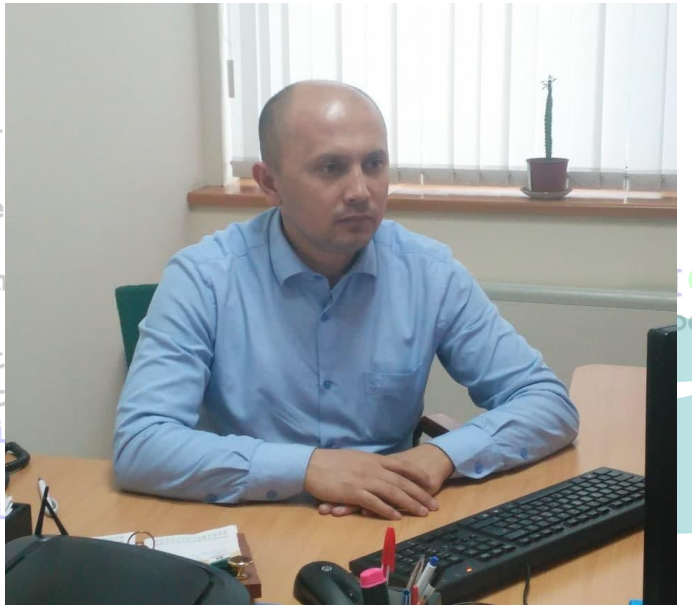
SUN’IY INTELLEKT ASOSLARI

SIA1046

02 NumPy kutubxonasi bilan ishlash



Raxmanov Qurbon Sodikovich
t.f.n., raxmanov@gmail.com

Reja

1. Data analysisda NumPy kutubxonasi.
2. Numpyda massivlar bilan ishlash.
3. Ma’lumotlar turi. Indekslash va Kesish.
4. Universal funksiyalar. Mantiqiy shart operatori.
5. Arifmetik amallar. Matematik va statistik amallar.
6. Tartiblash (Sorting).
7. Takrorlanmas va boshqa amallar.
8. Fayllar bilan ishlash.
9. Chiziqli Algebra.

NumPy kutubxonasi

Numpy, o‘zining to‘g‘ri nomi bilan "Numerical Python" deb tarjima qilinadi.

Python dasturlash tilida matematik amaliyotlarini amalga oshirish uchun keng qo‘llaniladigan bir kutubxona.

NumPy orqali ma’lumotlar matritsalari va vektorlarini boshqarish, matematik amaliyotlarini amalga oshirish, statistik ma’lumotlarni hisoblash va boshqalar kabi amaliyotlarni osonlashtirish mumkin. Numpy Pythonning asosiy kutubxonalardan biri hisoblanadi va amaliyotlarni tez va samarali bajarishga imkon beradi.

Nima uchun NumPy?

Kuchli N-o‘lchovli massivlar – Vektorlashgan hisoblash.

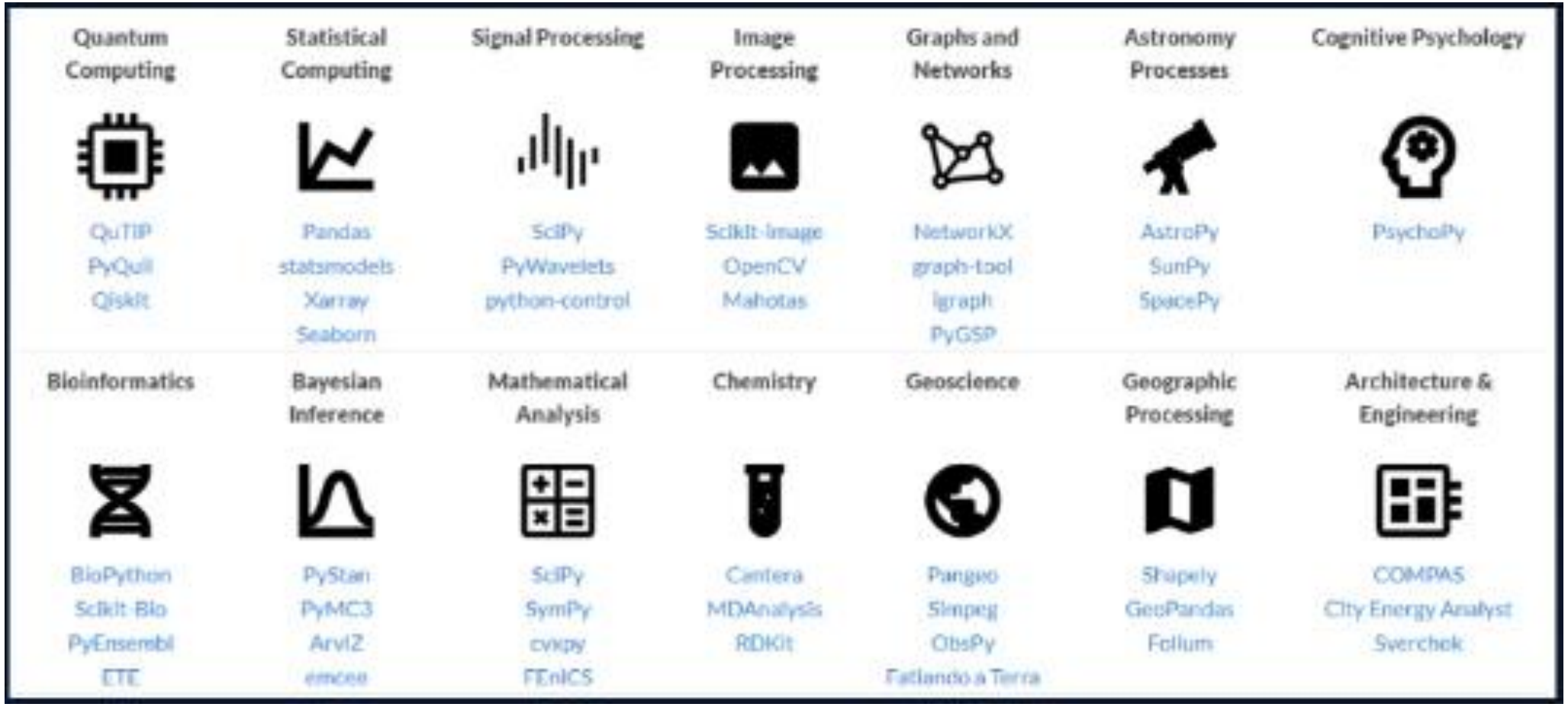
Raqamli hisoblash uskunalari – Matematik funksiyalar, Chiziqli algebra, Fourier transorms va boshqalar.

Ishlashdagi tezligi – NumPy kutubxonasining asosi C dasturlash tilida yaratilgan.

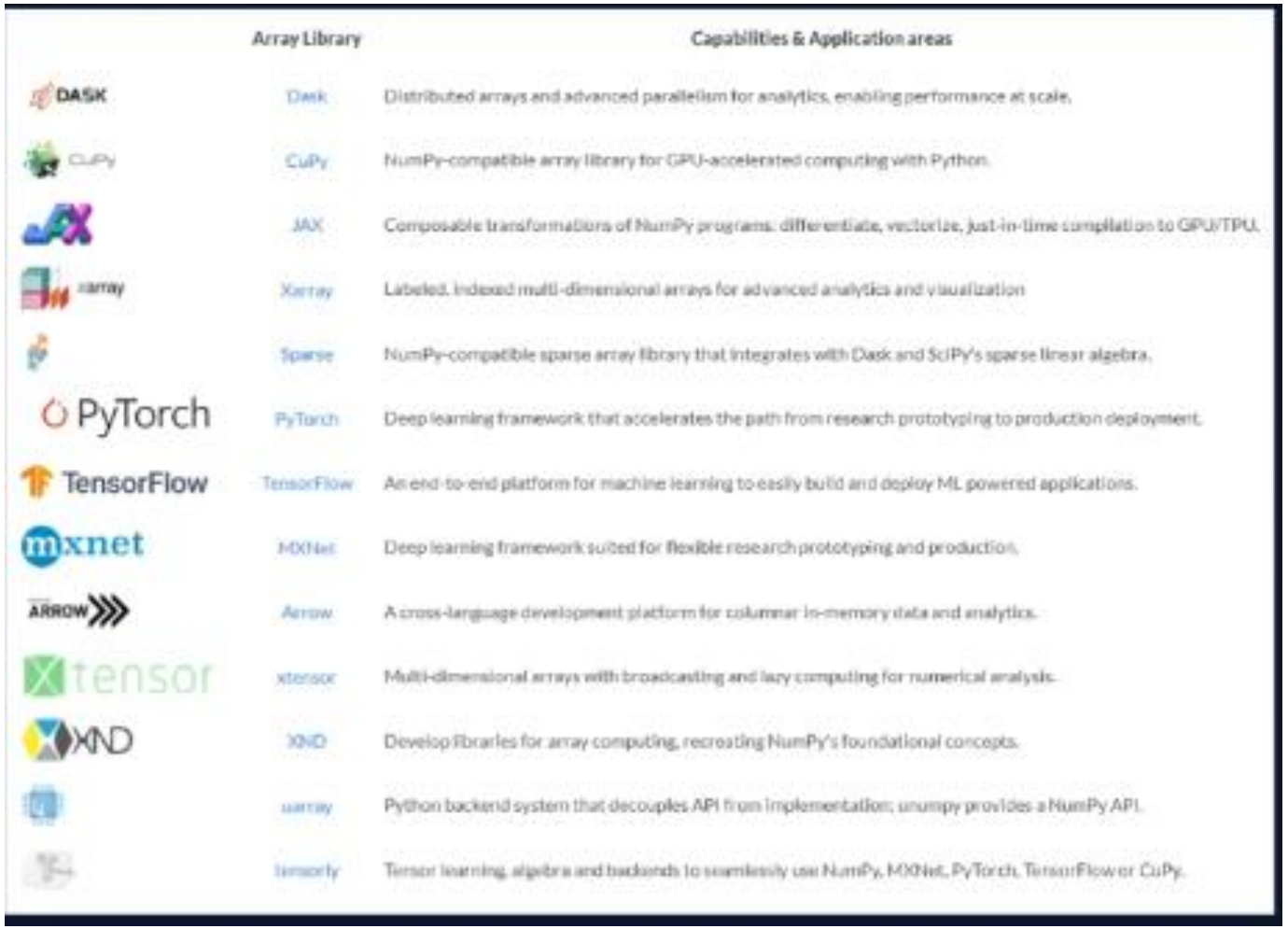
Ishlatishga oson – Tajribadan qat’iy nazar, har qanday inson osongina qo‘llay olishi mumkin.

Ochiq manbaaligi (Open Source) – BSD litsenziyasi asosida yaratilganlig bo‘lib, doimo bepul bo‘lishi ta’minlanadi.

NumPy qo‘llanilishi (Ilm - fan)



NumPy qo‘llanilishi (massivlar kutubxonasi)



Array Library		Capabilities & Application areas
DASK	Dask	Distributed arrays and advanced parallelism for analytics, enabling performance at scale.
CuPy	CuPy	NumPy-compatible array library for GPU-accelerated computing with Python.
JAX	JAX	Composable transformations of NumPy programs: differentiate, vectorize, just-in-time compilation to GPU/TPU.
xarray	Xarray	Labeled, indexed multi-dimensional arrays for advanced analytics and visualization
	Sparse	NumPy-compatible sparse array library that integrates with Dask and SciPy's sparse linear algebra.
PyTorch	PyTorch	Deep learning framework that accelerates the path from research prototyping to production deployment.
TensorFlow	TensorFlow	An end-to-end platform for machine learning to easily build and deploy ML powered applications.
mxnet	MXNet	Deep learning framework suited for flexible research prototyping and production.
ARROW	Arrow	A cross-language development platform for columnar in-memory data and analytics.
xtensor	xtensor	Multi-dimensional arrays with broadcasting and lazy computing for numerical analysis.
XND	XND	Develop libraries for array computing, recreating NumPy's foundational concepts.
	uarray	Python backend system that decouples API from implementation; unumpy provides a NumPy API.
	tensorly	Tensor learning, algebra and backends to seamlessly use NumPy, MXNet, PyTorch, TensorFlow or CuPy.

NumPy qo‘llanilishi (DATA SCIENCE)

NumPy lies at the core of a rich ecosystem of data science libraries. A typical exploratory data science workflow might look like:

Extract, Transform, Load: Pandas, Intake, PyJanitor
Exploratory analysis: Jupyter, Seaborn, Matplotlib, Altair
Model and evaluate: scikit-learn, statsmodels, PyMC3, spaCy
Report in a dashboard: Dash, Panel, Voila

For high data volumes, Dask and Ray are designed to scale. Stable deployments rely on data versioning (DVC), experiment tracking (MLFlow), and workflow automation (Airflow and Prefect).

NumPy qo‘llanilishi (MACHINE LEARNING)

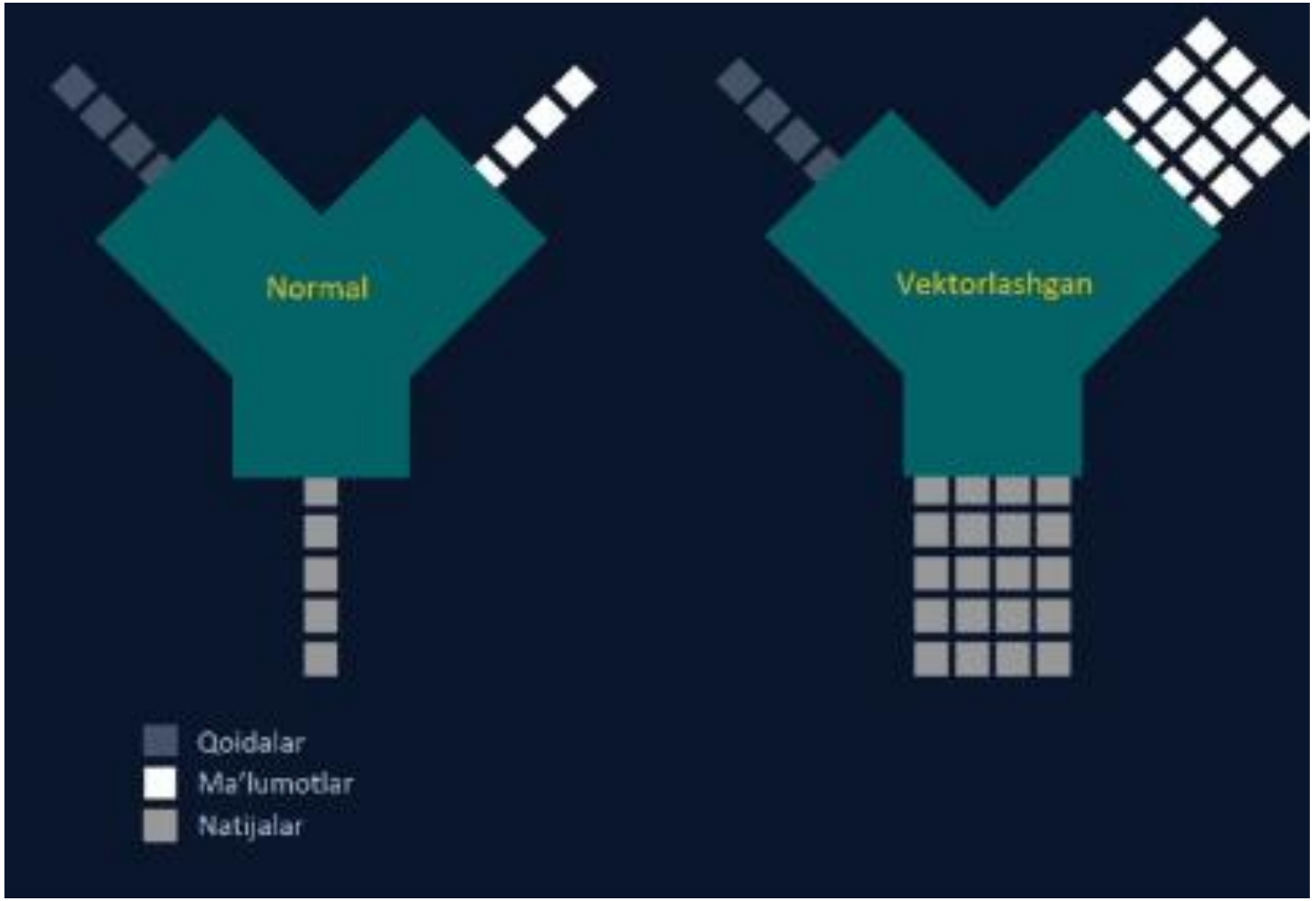
Source: Google AI Blog

NumPy forms the basis of powerful machine learning libraries like scikit-learn and SciPy. As machine learning grows, so does the list of libraries built on NumPy. TensorFlow's deep learning capabilities have broad applications — among them speech and image recognition, text-based applications, time-series analysis, and video detection. PyTorch, another deep learning library, is popular among researchers in computer vision and natural language processing. MXNet is another AI package, providing blueprints and templates for deep learning.

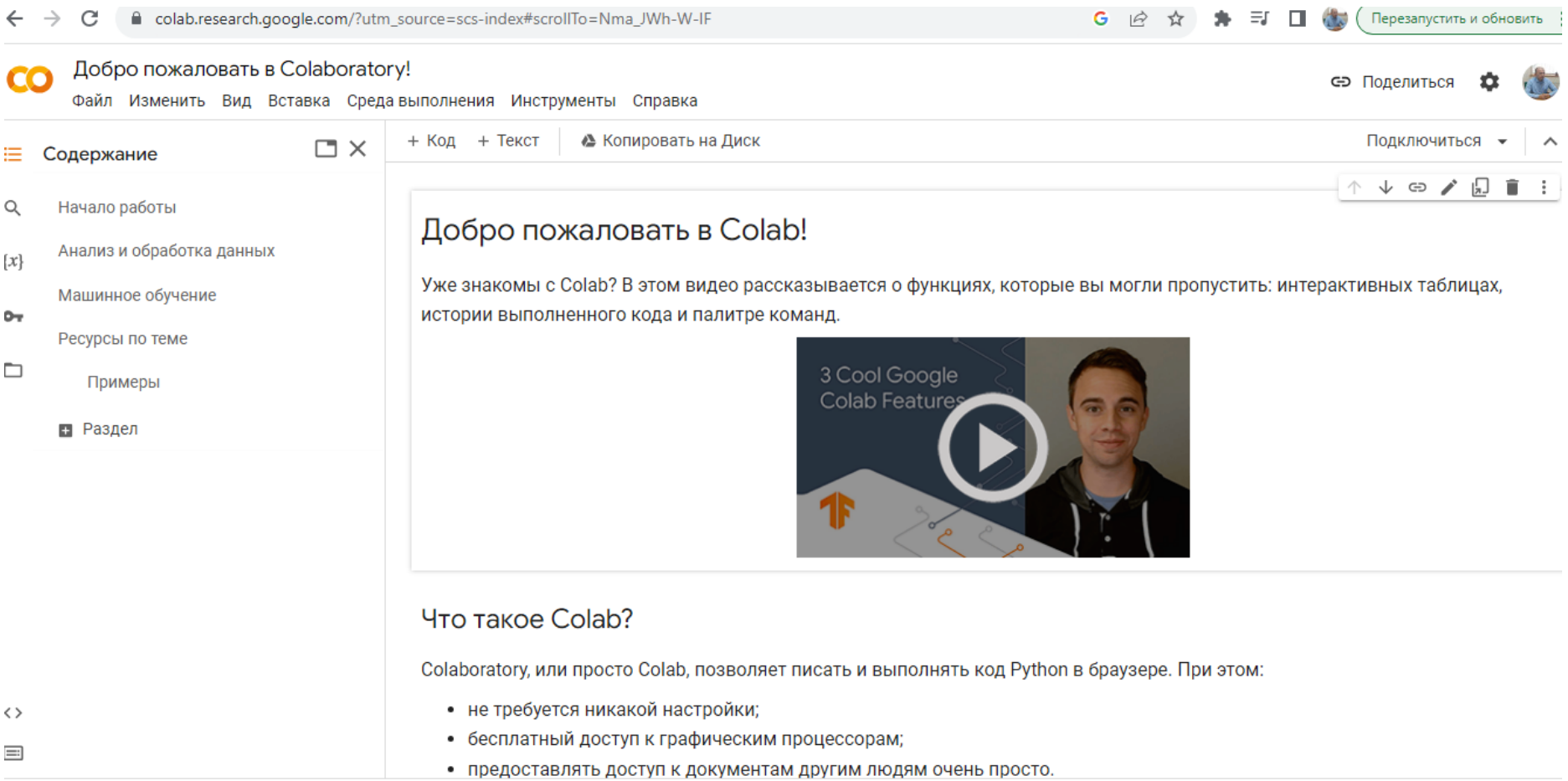
Statistical techniques called ensemble methods such as binning, bagging, stacking, and boosting are among the ML algorithms implemented by tools such as XGBoost, LightGBM, and CatBoost — one of the fastest inference engines. Yellowbrick and Eli5 offer machine learning visualizations.

VEKTORLASHGAN HISOBLASH

Vektorlashgan hisoblash – massiv ko‘rinishidagi ma’lumotlar to‘plamining barcha elementlari ustida bir vaqtning o‘zida hisoblash amallarini bajarish



Google Colabga kiramiz



NumPY da ishlash

```
#NumPy kutubxonasini chaqirib olish Shift+Enter bosiladi
import numpy as np
```

```
Python list bilan NumPy kutubxonasidagi massivlar (arraylar) hisoblashlari
orasidagi farqni ko'ramiz.
my_list = list(range(1000000)) # python list 0~99999 -->Normal
my_array = np.array(range(1000000))
# numpy array(massiv) 0~99999 --> Vektorlashgan
```

```
%time for _ in range(10): [x*2 for x in my_list]
# Normal #massiv har bir elementi 2 ga ko'paytirilayapdi
```

```
%time for _ in range(10): my_array*2
# Vektorlashgan #massiv har bir elementi 2 ga ko'paytirilayapdi
```

NumPY da ishlash

numPY_1.ipynb

Файл Изменить Вид Вставка Среда выполнения Инструменты Справка Изменения сохранены

Комментировать

+ Код + Текст

Python list bilan NumPy kutubxonasidagi massivlar (arraylar) hisoblashlari orasidagi farqni ko'ramiz.

```
[12] my_list = list(range(1000000)) # python list 0~99999 -->Normal
     my_array = np.array(range(1000000)) # numpy array(massiv) 0~99999 --> Vektorlashgan
```

```
[21] %time for _ in range(10): [x*2 for x in my_list] # Normal #massiv har bir elementi 2 ga ko'paytirilayapdi
```

```
CPU times: user 604 ms, sys: 92.7 ms, total: 697 ms
Wall time: 701 ms
```

```
[22] %time for _ in range(10): my_array*2 # Vektorlashgan #massiv har bir elementi 2 ga ko'paytirilayapdi
```

```
CPU times: user 13 ms, sys: 0 ns, total: 13 ms
Wall time: 12.5 ms
```

```
[25] 701/12.5 # ikkalasi orasidagi farq
```

```
56.08
```

Demak, vektorlashgan massiv 56.08 marta tez ishlar ekan

N-o‘lchamli massiv (array)larga ishlov berish

NumPyda massiv yaratishning eng oson yo‘llaridan biri array funksiyasi yordamida amalga oshiriladi. Ushbu funksiya har qanday turdagi ketma-ket ma’lumotlarni qabul qilib, yangi Numpy arrayiga o‘giradi.

```
data1 = [3.5, 5, 6, 2] # list
arr1 = np.array(data1) # array1
arr1 #Shift + Enter
```

```
array([3.5, 5. , 6. , 2. ])
```

```
data2 = (4, 5, 3.2, 7) # tuplam
arr2 = np.array(data2) # array
arr2
```

```
array([4., 5. , 3.2 , 7. ])
```

N-o‘lchamli massiv (array)larga ishlov berish

NumPyda massiv yaratishning eng oson yo‘llaridan biri array funksiyasi yordamida amalga oshiriladi. Ushbu funksiya har qanday turdagi ketma-ket ma’lumotlarni qabul qilib, yangi Numpy arrayiga o‘giradi.

```
[28] data1 = [3.5, 5, 6, 2] # list
     arr1 = np.array(data1) # array1
     arr1
```

```
array([3.5, 5. , 6. , 2. ])
```

```
[31] data2 = (4, 5, 3.2, 7) # tuplam
     arr2 = np.array(data2)
     arr2
```

```
array([4. , 5. , 3.2, 7. ])
```

Massiv o‘lchami

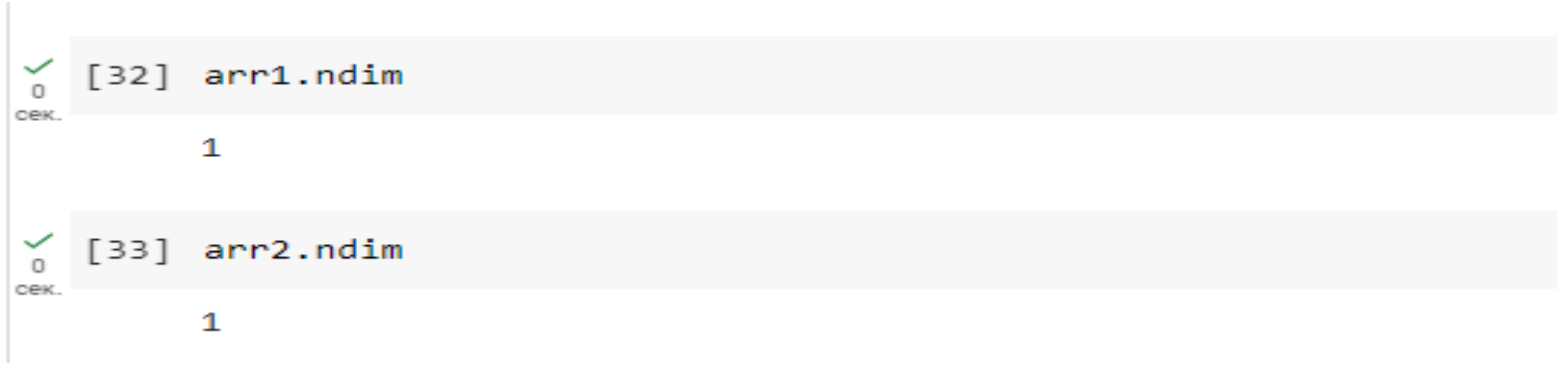
Massiv o‘lchamini aniqlash ndim orqali amalga oshiriladi

```
arr1.ndim # array1 ning o‘lchami
```

```
1
```

```
arr2.ndim # array2 ning o‘lchami
```

```
1
```



Ikki va undan katta o‘lchamli massivlar

```
data3 = [[1, 2, 3, 4], [5, 6, 7, 8]] # list ichidagi list (nested list)
arr3 = np.array(data3) # array3
arr3.ndim # array3 ning o‘lchami
```

```
2
```

```
data3 = [[1, 2, 3, 4],[5, 6 , 7 , 8]] # tuplam
arr3 = np.array(data3)
arr3
```

```
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

```
[39] arr3.ndim
```

```
2
```

```
[40] data4 =((1, 2, 3, 4),(5, 6 , 7 , 8)) # tuplam
     arr4 = np.array(data4)
     arr4
```

```
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

```
[41] arr4.ndim
```

```
2
```

NumPy shape va size metodlari

shape metodi massivning qator va ustunlar sonini ko‘rsatsa, size esa massivlardagi elementlar sonini chiqaradi

```
arr3.shape # qator va ustunlar sonini ko‘rsatadi (q, u)
```

```
(2,4)
```

```
arr5.shape # qator va ustunlar sonini ko‘rsatadi (q, u)
```

```
(3,4)
```

```
arr3.size # array3 ning elementlar soni
```

```
8
```

```
arr5.size # array3 ning elementlar soni
```

```
12
```

NumPy shape va size metodlari

```
[46] arr3.shape # qator va ustunlar sonini ko'rsatadi (q, u)
```

```
(2, 4)
```

```
[47] arr5.shape
```

```
(3, 4)
```

```
[48] arr3.size
```

```
8
```

```
[49] arr5.size
```

```
12
```

N-o‘lchamli massivlar yaratishning boshqa usullari

zeros va ones funksiyalari yordamida massivlar yaratish

```
def add5(x):
    return x+5

def dotwrite(ast):
    nodename = getNodename()
    label=symbol.sym_name.get(int(ast[0]),ast[0])
    print '    %s [label="%s' % (nodename, label),
    print '"];'
    children = []
    for n, child
        children.ap
    print '    %s
    for name in children:
        print '%s' % name,
```

E’tiboringiz uchun rahmat!

Raxmanov Qurbon Sodikovich
t.f.n., raxmanov@gmail.com