Our Partners

Microsoft

Platinum

Quest

DBPLUS better performance

Enterprise Training Center

Gold

IDERA

oh22information services

SOLISYON

SolidQ Think Big. Move Fast.

trivadis makes IT easier.

Bronze

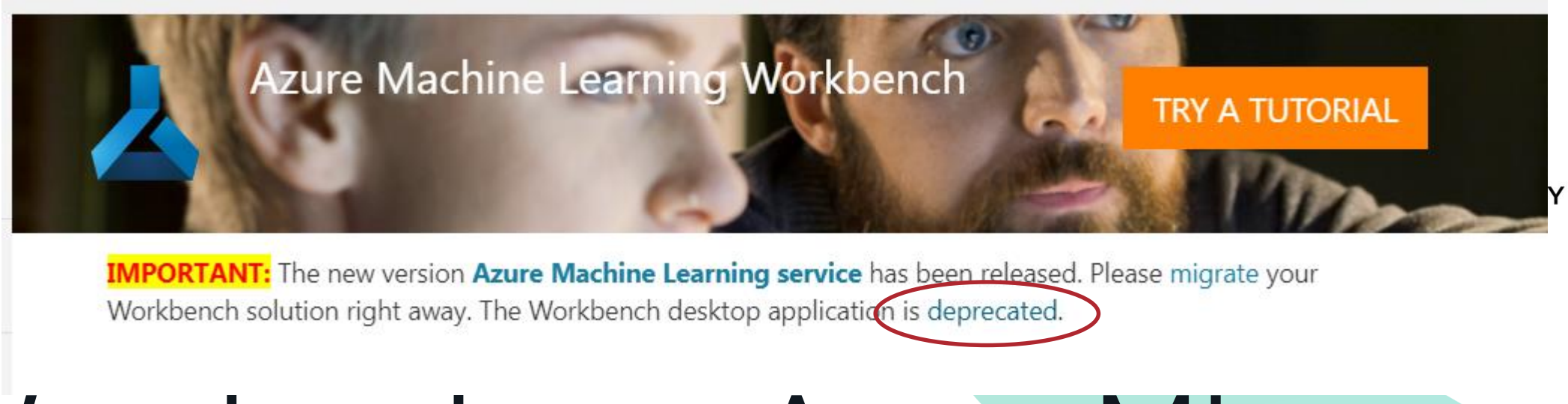
PASS

pmOne Analytics

UBIDO AN ACP GROUP COMPANY

GRZ IT Center GmbH

Mario Schnalzenberger



Woodworks on Azure ML Workbench

Fragen?

Danke

Mario Schnalzenberger

Working with Azure ML services

The Speaker

- Informatiker, Statistiker und Volkswirt
- (nicht mehr so viel) Forscher
  - Forschung im Bereich Gesundheit, Alterung, Pensionen und vielem mehr
  - Veröffentlichungen in Klinischer Forschung, Economics und Econometrics
- bei cubido unterstütze ich Kunden im Bereich
  - DWH und Business Intelligence
  - Predictives in Richtung Industrie 4.0 und Marketing Intelligence
  - Big Data und verwandten Themen
  - SQL Server, Cubes, MDX, R, C#, SAP Infinite Insights, MCSA, MCSE BI, uvm.
- m.schnalzenberger@cubido.at

CUBIDO
AN ACP GROUP COMPANY

Python or R

Basics in Azure ML services

Language Preference for Python but

Familiar data science tools

- Jupyter Notebooks
- Visual Studio Code
- PyCharm (was my first choice coming from R)

First experiment

- What is an Experiment?
- What is Azure ML services useful for?

- Let's calculate PI – why – just because I found the code …

```
run = experiment.start_logging()

pi_counter = 0
n_iter = 1000000

# Log total number of iterations
run.log("Number of iterations",n_iter)

for i in range(1,n_iter):
    # Monte Carlo step to update estimate
    x = random.random()
    y = random.random()
    if x*x + y*y < 1.0:
        pi_counter += 1
    pi_estimate = 4.0*pi_counter / i

    # Log convergence every 10000 iterations
    if i%10000==0:
        error = math.pi-pi_estimate
        run.log("Pi estimate",pi_estimate)
        run.log("Error",error)

# Log final results
run.log("Final estimate",pi_estimate)
run.log("Final error",math.pi-pi_estimate)

# Write file containing pi value into run history
with open("pi_estimate.txt","wb") as f:
    pickle.dump(str(pi_estimate),f)
run.upload_file(name = 'outputs/pi_estimate.txt', path_or_stream = './pi_estimate.txt')

# Complete tracking and get link to details
run.complete()
print("Run completed")
```

```
run = experiment.start_logging()

p  ounter = 0
n_i    = 1000000

# Log t   number of iterat
run.log("N   er of iteration

for i in range(   iter):
    # Monte Carl   ep to up
    x = random.rand  )
    y = random.random
    if x*x + y*y < 1.0:
        pi_counter += 1
    pi_estimate = 4.0*pi_cou

    # Log convergence every
    if i%10000==0:
        error = math.pi-pi_e
        run.log("Pi estimate
        run.log("Error",erro

# Log final results
run.log("Final estimate" pi
run.log("Final err       .p

# Write file containing pi value into run history
with open("pi_estimate.txt","wb") as f:
    pickle.dump(str(pi_estimate),f)
run.upload_file(name = 'outputs/pi_estimate.txt', path_or_stream = './pi_estimate.txt')

# Complete tracking and get link to details
run.complete()
print("Run completed")
```

Active Filters

No child runs

MAX NUMBER OF ITERATIONS

1000000

AVG FINAL ESTIMATE

3.143

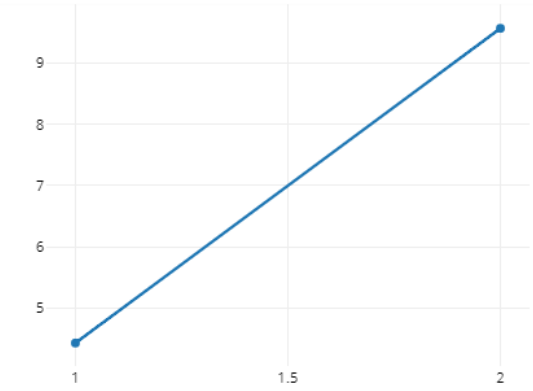
MAX FINAL ERROR

-0.0002625

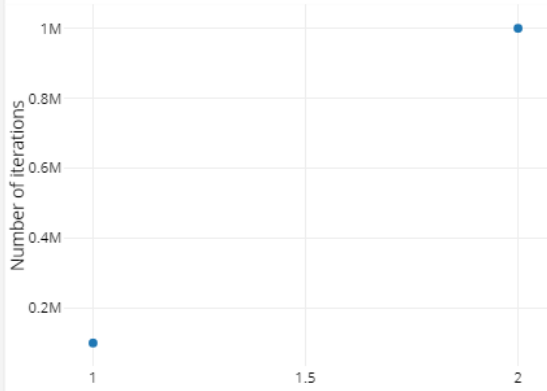
RUNNING

0

DURATION (SECONDS)



NUMBER OF ITERATIONS



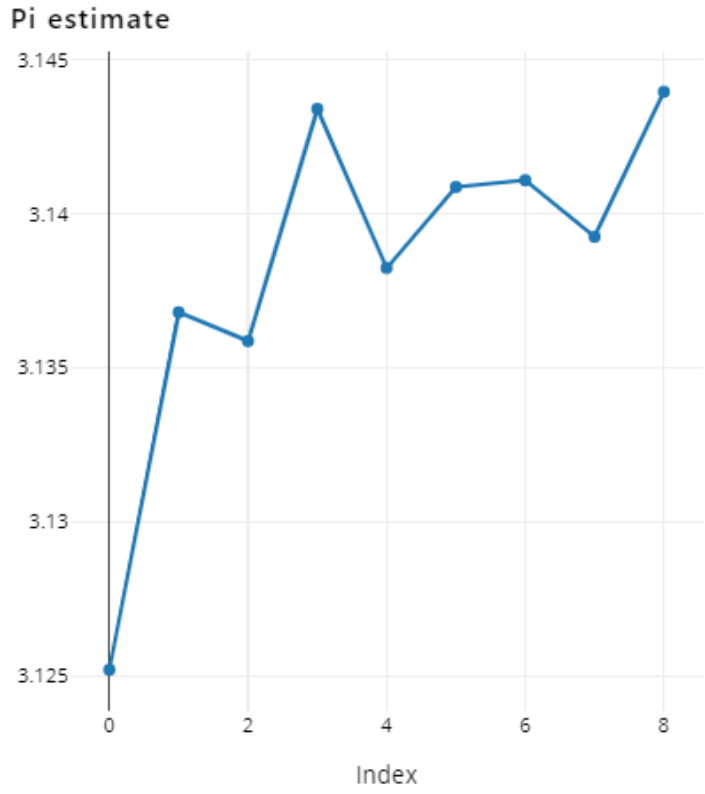
RUN NUMBER	STATUS	DURATION (SECONDS)	NUMBER OF ITERATIONS	FINAL ESTIMATE	FINAL ERROR
2	Completed		1000000	3.142	-0.0002625
	pleted	4.432	100000	3.144	-0.002359

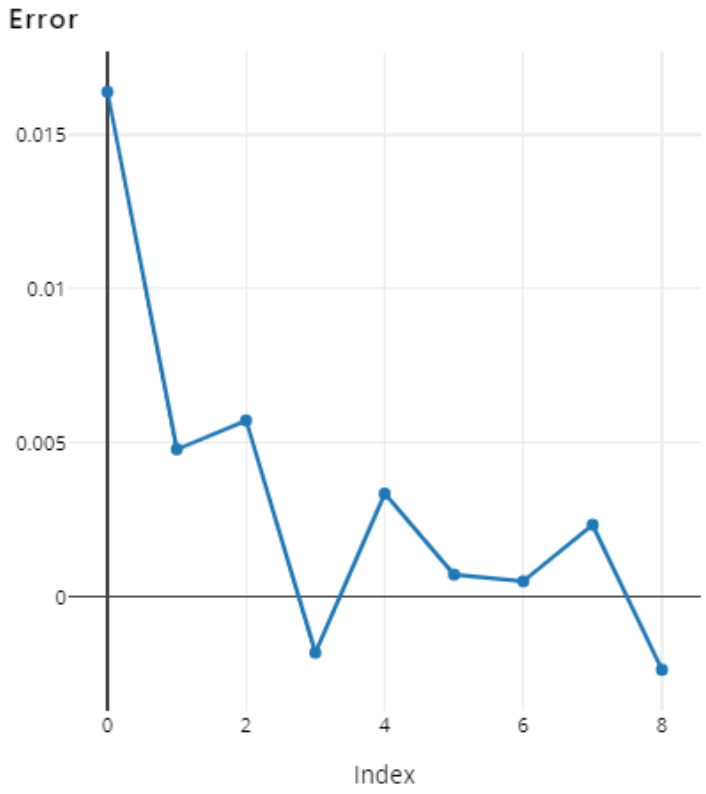
UBIDO
AN ACP GROUP COMPANY

Detailed experiment information

```
# Log convergence every 1
if i%10000==0:
    error = math.pi-pi_es
    run.log("Pi estimate"
    run.log("Error",error
```

CHARTS





CUBIDO
AN ACP GROUP COMPANY

Demotime

Scoring … some more basics before

Model is simply a file - or a folder of files - that model management service tracks and versions. (trained model – pickle).

Image is a combination of model, a Python scoring script and Python libraries. (It is a self-contained unit that can be deployed as a service.)

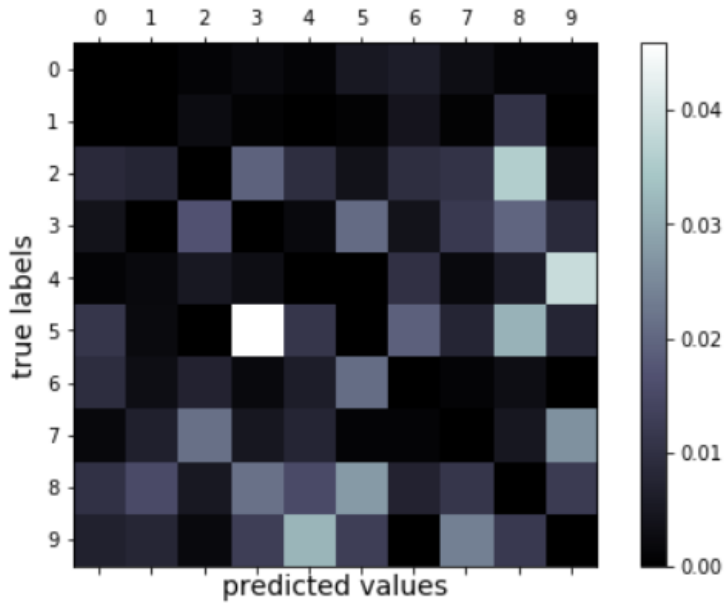
Service is the image running on a compute. The service can be called!

Demotime

Demotime
Image Classification using MNIST

In [11]:
```
# normalize the diagnal cells so that they don't overpower the rest of the cells when visualized
row_sums = conf_mx.sum(axis=1, keepdims=True)
norm_conf_mx = conf_mx / row_sums
np.fill_diagonal(norm_conf_mx, 0)

fig = plt.figure(figsize=(8,5))
ax = fig.add_subplot(111)
cax = ax.matshow(norm_conf_mx, cmap=plt.cm.bone)
ticks = np.arange(0, 10, 1)
ax.set_xticks(ticks)
ax.set_yticks(ticks)
ax.set_xticklabels(ticks)
ax.set_yticklabels(ticks)
fig.colorbar(cax)
plt.ylabel('true labels', fontsize=14)
plt.xlabel('predicted values', fontsize=14)
plt.savefig('conf.png')
plt.show()
```



Quite astonishing results

CUBIDO
AN ACP GROUP COMPANY

Steps

1. Work in Python (Jupyter, PyCharm, VS Code, …. ?) as usual
2. Find valid Models => if too much for your workstation make usage of compute clusters (including Spark, Tensorflow, and others)
3. Version the models and build „services“, i.e. valid scoring strategies as APIs
4. Version the services if necessary

Wrap up

- The **important** thing is the **Service**
- The deployment cycle is part of the game.
- A tool to fully integrate the „data scientist“ in versioning.

Und jetzt wirklich ....

Fragen?

Danke