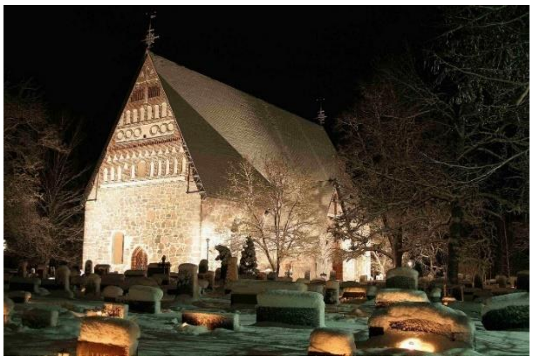
Zen and the Art of Machine Learning:

Born of Larvaus

by Marko Peltojoki

Introduction to motivation



Once upon a time there was a little boy living in the tiny rural village far far away in the small insignificant country, which nobody has heard of. That tiny rural village was once significant, like 500 years ago when it got its medieval stone church, one of the biggest in that insignificant country at that time. By the legend it was built by two giants alone. Ever since it has been immersed into oblivion so deep, that nobody knows where that tiny rural village is located. It is nowhere, and yet it is found all over the world. Once significant and now in oblivion. So, we better forget that tiny rural village, as this is a story about the boy and not about the village. Actually, in the end this is not a story about the boy either, but about the journey to your inner zen garden.

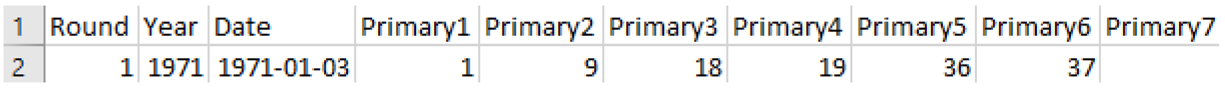
The little boy was living the ordinary life as any little boy in the tiny rural village, chasing the cows and horses, carrying the water bucket he filled with fresh cold water from the well into the small red house he was living in, exploring the forbidden places in surroundings, like the cold mortuary at the basement of the bell tower, or climbing up to the bell tower which stairs were blocked by the high fence, or ringing the church bell, not loud but just and just to get excited if someone heard, and finally find the signature of his teacher and principal of the village school he was attending at the top of the bell tower, which proved he was not the only naughty kid in that tiny rural village. It seems to have been village tradition to sneak into the bell tower and leave your mark there.

As the little boy was born in a poor family, he knew what it was like to live on a shoestring budget. He could only dream of wealthy life. He earned some small amounts of money by doing this and that, mainly by collecting empty bottles and returning them to the grocery store, as there was refund system in place. Those small amounts of money he spent mainly on salty liquorice and sugar candy, but every now and then he invested in local lottery game. The local lottery game had the greatest cash prizes at that time. There was also horse race game, but it did not yield so great amounts of money and you had to know the horses by name. The friends of the little boy knew all the horses even by looking the picture of the horse race, as they have born in the middle of the horse stables and breathed the horse fumes since they were born. One of them became the great horse driver when he grew older. But the little boy has not born in that tiny rural village and did not know all the horses by the name or by the look. Actually, he scared the horses which were huge compared to the little boy. Once he was helping on the horse stables by taking the horse to its stall, when it suddenly stopped and stood still on the foot of the little boy. Oh, it hurt so bad, but as there were queue of horses, the only traffic jam ever in that tiny rural village, he could not do anything else than wait, silently as he was raised to not complaint even if it hurts. Ever since he kept away of horses. So, he chose the local lottery game to become rich, which he has been playing every now and then ever since, without great prizes though.

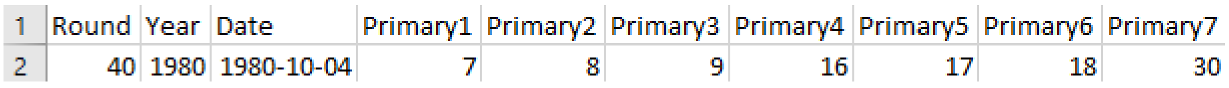
Business problem

Fast forward several decades and the little boy has grown up to the middle age man. As the investments to the local lottery has had negative revenue stream ever since, nothing major but negative anyway, he decided to find the way to get even, or at least get the advantage over the regular player. Currently the odds of regular player to get the grand prize in one round by playing only one row is mathematically 1 out of 18 643 560, as there are 40 numbers to choose from and only 7 numbers are drawn on the round. There are also some secondary numbers drawn which are omitted as not important, because to win the grand price there need to be all the 7 primary numbers correct. How to improve the odds?

First, you need the dataset of course. The bigger the better. Luckily the colleague had collected 60% of the dataset years earlier, so there was missing only 22 years of data, which were found on the local lottery provider's web site. Some web scraping, cleaning of the data, sorting it out, visualizing it and you had a file you could call the dataset. Dataset which needed to be split as there were fluctuation of drawn numbers as well as total numbers to choose from. The optimal split was done around at the end of the year 1980.

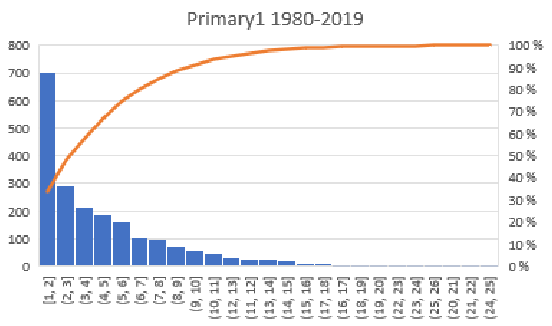


1	Round	Year	Date	Primary1	Primary2	Primary3	Primary4	Primary5	Primary6	Primary7
2	1	1971	1971-01-03	1	9	18	19	36	37	



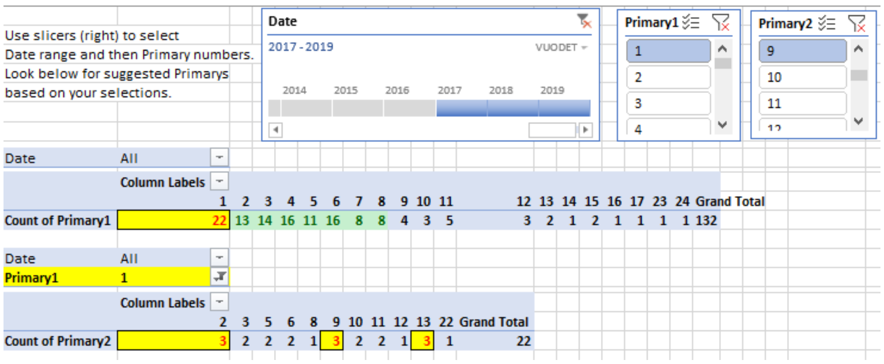
1	Round	Year	Date	Primary1	Primary2	Primary3	Primary4	Primary5	Primary6	Primary7
2	40	1980	1980-10-04	7	8	9	16	17	18	30

After analyzing the 38 years of local lottery data, especially sorting the numbers of winning row to ascending order and looking the distribution of each position (smallest number drawn, second smallest…,…, largest number drawn) over the history, it was revealed that rather than having odds for first (in this context the smallest) number drawn as of 7/40 the odds were like 7/26 (when smallest number distribution was from number 1 to number 26). After calculation of all the distributions and odds in each position, was the odds more like 1 out of 3 858 998, which is 4,8 times better than the regular odds. Not bad at all. Any business would be interested if you could improve their odds 4-5 folds! At the same time, we need to keep in mind that this is the game of a great fortune, still. Or you just fill in all those 3,9 million rows then.



Further analyzing the distributions, like if the first smallest number was for example 1, what were the most common numbers for the second smallest number during the last three years (2, 9 and 13), and then if chosen 9 what was the next, etc.? So exciting to look those patterns emerge from the dataset. Was there mention of "patterns"? Sounds like the dataset is ready for the Machine Learning model then.

Actually, to save some compute cycles we could just stop here, fiddle the numbers in the Excel and have the same odds. But then, where is all the fun, all the issues, all the coding, all the CI/CD hassle, all the brain melting out of your ears and nose? We would just miss all of that, no pushing oneself out of comfort zone, no learning at all! No way! So, let’s continue to the Machine Learning la-la-land.



Use slicers (right) to select
Date range and then Primary numbers.
Look below for suggested Primarys
based on your selections.

Date	All

	Column Labels																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	23	24	Grand Total
Count of Primary1	22	13	14	16	11	16	8	8	4	3	5	3	2	1	2	1	1	1	1	132

Date	All
Primary1	1

	Column Labels											
	2	3	5	6	8	9	10	11	12	13	22	Grand Total
Count of Primary2	3	2	2	2	1	3	2	2	1	3	1	22

Rise of the Machines

When the exercise was semi-serious and the Machine Learning model could not do much, the dice was cast for Automated Machine Learning to save time and have low-code/no-code solution. Hey! The person at the front row may stop ripping the pants immediately. Just calm down and read on.

As happens, sometimes the planets are aligned, and there was Azure Automated Machine Learning graphical user interface available on preview for our Machine Learning hero to utilize. Oh, yeah! No code! You may follow the setup steps here, https://docs.microsoft.com/en-us/azure/machine-learning/service/how-to-create-portal-experiments .

Now follows the Caveat Emptor (Let the buyer beware), or few of them. The Azure Automated Machine Learning do wonderful stuff and pretty fast but take care of the following pit falls.

Be aware, that after You have run the experiments and created the models to be deployed, delete the compute used under Compute in your Machine Learning service workspace. Yes, no need to keep it running to add up your subscription bill for nothing.

Ok, now deep breathe. Are You ok with that? Can we continue? Are You sure?

That was one of the wonders which took some time to realize when seeing the money drain and got the courage to finally delete the compute.

Second, do not select blindly the “best” experiment for your model to be deployed. Look the details and metrics inside of each promising experiment. Before that You really must know Your data.

Why is that? Best is the best, right? No!

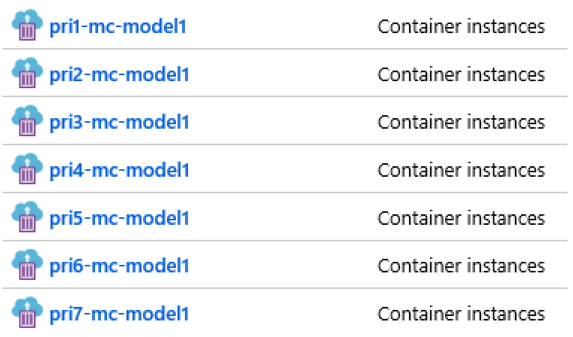
With selected dataset there was one number with overweight, number 1, when selecting the first (smallest) number (naturally as it was the smallest one and cannot exist also as second smallest number, like number 2 can be the smallest or second smallest on the round, number 3 can be the smallest, second or third smallest on the round, etc.). Guess what numbers the “best” experiment predicted. Only number 1, even there were 25 other numbers to choose from and handful with high probability. So, have a closer look what each experiment predicts, know your data thoroughly and select wisely the model to be deployed to avoid any bias in predictions.

When predicting the first number, called Primary1, the Round (week number) and the Year were only variables to be known. To avoid any overlaps and conflicts, like four number 9s in a row, for the second number, called Primary2, there is Round, Year and Primary1 as variables, so we have a feedback loop there. And for the third number, called Primary3, You feed Round, Year, Primary1 and Primary2. And so forth.

At the end there is seven models to be deployed, one for each Primary. Lot of compute power for tiny little task like that when each model has its individual container. There might be a way to use only one container, but that subject still needs further studies and higher education.

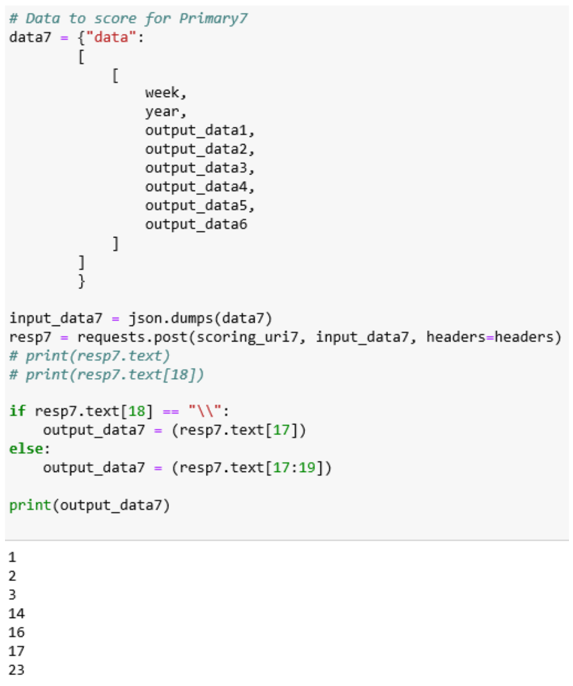
Third hint, if You do not need to make any predictions, shut down the container instances where the model is located as web service. You may find the containers under the resource group used.

This was another thing which took some time to realize, how to shut down the container instance as there is no such option under your Machine Learning service workspace, which would be the most logical place to look at first.



pri1-mc-model1	Container instances
pri2-mc-model1	Container instances
pri3-mc-model1	Container instances
pri4-mc-model1	Container instances
pri5-mc-model1	Container instances
pri6-mc-model1	Container instances
pri7-mc-model1	Container instances

This far it was quite quick to get. Then started the slow part. How to consume those models? Especially when you have feedback loop in there. Luckily the thorough instructions were provided for several programming languages and even for PowerBI: https://docs.microsoft.com/en-us/azure/machine-learning/service/how-to-consume-web-service . Kick up the Jupyter notebook server (or VM) or use Notebooks.azure.com if using Python notebook, the easiest way to test the models.



```
# Data to score for Primary7
data7 = {"data":
        [
            [
                week,
                year,
                output_data1,
                output_data2,
                output_data3,
                output_data4,
                output_data5,
                output_data6
            ]
        ]
        }

input_data7 = json.dumps(data7)
resp7 = requests.post(scoring_uri7, input_data7, headers=headers)
# print(resp7.text)
# print(resp7.text[18])

if resp7.text[18] == "\\":
    output_data7 = (resp7.text[17])
else:
    output_data7 = (resp7.text[17:19])

print(output_data7)
```

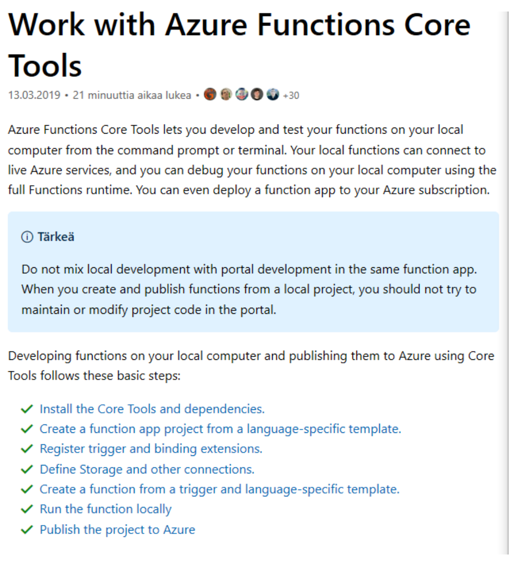
```
1
2
3
14
16
17
23
```

Yeah, works as expected. Neat, right? But, what next? How to take it a bit further? Would it be cool to have the results in your web browser instead of Jupyter notebook? Let that sink in first. Web browser, mobility, accessibility, tremendous business opportunities, drooooolllll….!

Show me the honey

As the web development was a bit unexplored waters for the Machine Learning hero, took it some time to find the correct approach. Azure Functions looked promising, but after getting familiar with all the pre-requisites and capabilities, the Azure App Service was easier to implement.

https://docs.microsoft.com/en-us/azure/app-service/containers/quickstart-python



Work with Azure Functions Core Tools

13.03.2019 • 21 minuuttia aikaa lukea • +30

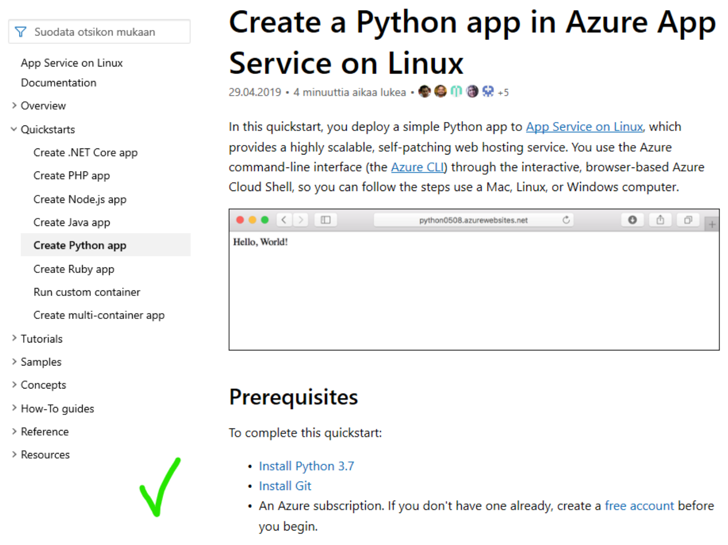
Azure Functions Core Tools lets you develop and test your functions on your local computer from the command prompt or terminal. Your local functions can connect to live Azure services, and you can debug your functions on your local computer using the full Functions runtime. You can even deploy a function app to your Azure subscription.

ⓘ Tärkeä

Do not mix local development with portal development in the same function app. When you create and publish functions from a local project, you should not try to maintain or modify project code in the portal.

Developing functions on your local computer and publishing them to Azure using Core Tools follows these basic steps:

- ✓ Install the Core Tools and dependencies.
- ✓ Create a function app project from a language-specific template.
- ✓ Register trigger and binding extensions.
- ✓ Define Storage and other connections.
- ✓ Create a function from a trigger and language-specific template.
- ✓ Run the function locally
- ✓ Publish the project to Azure



Create a Python app in Azure App Service on Linux

29.04.2019 • 4 minuuttia aikaa lukea • +5

In this quickstart, you deploy a simple Python app to App Service on Linux, which provides a highly scalable, self-patching web hosting service. You use the Azure command-line interface (the Azure CLI) through the interactive, browser-based Azure Cloud Shell, so you can follow the steps use a Mac, Linux, or Windows computer.

Prerequisites

To complete this quickstart:

- Install Python 3.7
- Install Git
- An Azure subscription. If you don't have one already, create a free account before you begin.

The quick start with Hello World example was quite useless to learn anything useful, but there was one hint on the code, word Flask. Bit further searching and world of Flask was revealed bit by bit. The most useful example was provided by Über, https://github.com/uber/Python-Sample-Application/blob/master/app.py . There was steep learning curve ahead, like what is the @app.route good for, how come Jupyter notebook code does not work directly as Python application, what are the user defined functions? Piece by piece the Python application was formed, it ran like a rat in local machine and did some useful stuff, much slower than the Jupyter notebook version but nevertheless it worked. It was time to upload the application to the Azure App Service.

Born of Larvaus

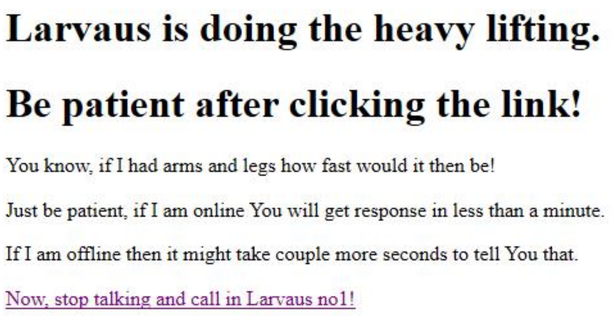
It was time to invent sticky name for the Azure App Service. As it was about Lottery, then it should be included somehow, also it was about guessing the numbers. The word “guess” is “arvaus” in the vocabulary of that insignificant country, so let the name be – Larvaus!

The name Larvaus was also close to term “larva” which is the active immature form of an insect, especially one that differs greatly from the adult and forms the stage between egg and pupa, e.g. a caterpillar or grub. Exactly descriptive term to tell where this application was. And there are cute images of Larva (tv series in South Korea) as well.

When the name was selected the fun was about to see the day light, right? Uhmmm, not quite, there was still some tweaking required.

Larvaus itself

The Larvaus application worked quite slowly in local test environment in its current (or should say in previous) form as it called all the Primary numbers at once to spew them out in the web page. As You might remember, there was the feedback loop to enter the previous number back to the next model as one of the variables. Crawling all the seven Machine Learning models through was time consuming, it took around 90 seconds to get all the numbers at once by the application while in Jupyter notebook it took something like 10 seconds. Might be the application code structure or then something else, but the Azure App Service has hard coded probe timeout as 31 seconds, which kicked in and hard. So, the application logic needed some workaround and now it calls only one number at the time, thus user needs to do the crawling through several pages. Just try it out by yourself. If errors out, then the author has run out of free credits and the Machine Learning containers are shut down. Just try few weeks later, then.



Larvaus is doing the heavy lifting.

Be patient after clicking the link!

You know, if I had arms and legs how fast would it then be!

Just be patient, if I am online You will get response in less than a minute.

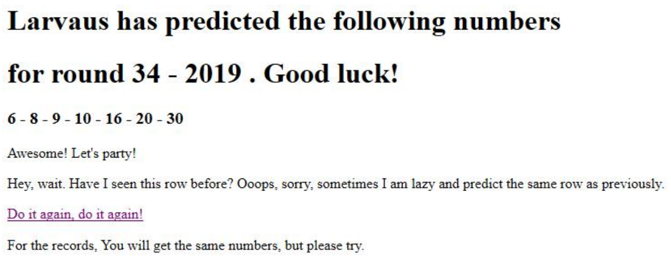
If I am offline then it might take couple more seconds to tell You that.

Now, stop talking and call in Larvaus no1!

https://larvaus.azurewebsites.net/

The summary page might timeout if not crawled through the first page but go ahead and try it out.

https://larvaus.azurewebsites.net/larvaus/



Larvaus has predicted the following numbers

for round 34 - 2019 . Good luck!

6 - 8 - 9 - 10 - 16 - 20 - 30

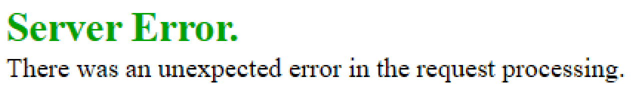
Awesome! Let's party!

Hey, wait. Have I seen this row before? Ooops, sorry, sometimes I am lazy and predict the same row as previously.

Do it again, do it again!

For the records, You will get the same numbers, but please try.

Azure App Services probe timeout 31 seconds not caught by the app most of the time, should throw Error.html page instead of Server Error when Machine Learning models are not online. Either the Machine Learning model webservice throws out different error than before which is not caught by the application or the App Service takes over before the application errors out itself.

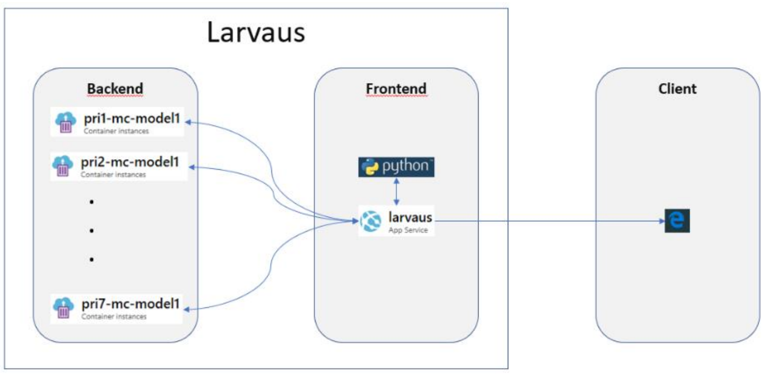


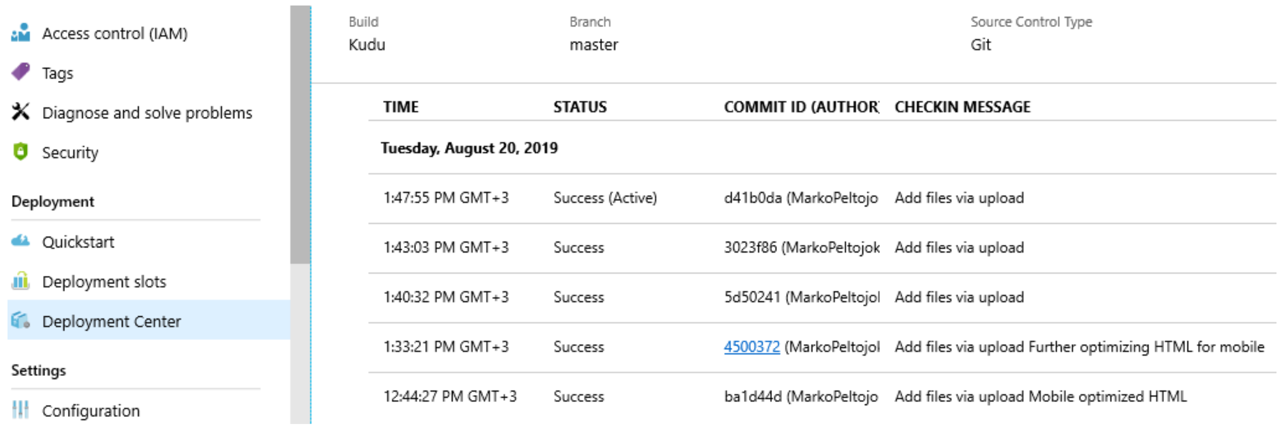
Server Error.

There was an unexpected error in the request processing.

One interesting finding was to see the models to predict the same numbers as in previous week, it might happen even three times in a row and several times during the year. Must be the super row then. Luckily for the next year the row changes, but the symptom repeats, even though not on the same rounds/weeks.

Most useful feature in Azure App Service was the Deployment Center, which integrates with GitHub repository and enables the continuous deployment. Depending on the App Service Plan tier there could be staging slots for dev/test before swapping code to production, but for the current purposes it was good enough to have direct deployment from GitHub repository. It took literally less than a minute to see the change in the Larvaus itself after commit to GitHub. Wow!





Build	Branch	Source Control Type
Kudu	master	Git

TIME	STATUS	COMMIT ID (AUTHOR	CHECKIN MESSAGE
Tuesday, August 20, 2019			
1:47:55 PM GMT+3	Success (Active)	d41b0da (MarkoPeltojo	Add files via upload
1:43:03 PM GMT+3	Success	3023f86 (MarkoPeltojok	Add files via upload
1:40:32 PM GMT+3	Success	5d50241 (MarkoPeltojol	Add files via upload
1:33:21 PM GMT+3	Success	4500372 (MarkoPeltojol	Add files via upload Further optimizing HTML for mobile
12:44:27 PM GMT+3	Success	ba1d44d (MarkoPeltojo	Add files via upload Mobile optimized HTML

You could deploy directly from Visual Studio Code as well, but it was preferred to upload the files into GitHub anyway, so why not deploy from there.

Be aware, commit application files directly to deployment repository root.

First try with GitHub deployment ended up to the brake of the application as there was Application folder which was uploaded to repository rather than the files and folders under the Application folder. You make mistakes, You learn, hopefully.

Conclusions

As mentioned in the beginning, this was about the journey itself, which matters most.

It does not matter if Larvaus does not yield a penny, money does not make You happy. They say it is more convenient to cry in Your Jaguar, than in overcrowded bus while stuck in the traffic jam. Who knows?

It does not matter if it hurt to get here where You stand now, there is always some horse standing in Your foot in some point of the time, always. It will go away eventually.

Have You left Your mark somewhere? It does not have to last 500 years, unless it is something huge and most likely carved into stone.

Have You explored the places unknown? It is recommended, so You might learn something new.

Do You know Your data well enough? Have You found any patterns there yet? Go and draft Your Machine Learning model using the tools and methods which fit for Your purpose. You will have Your share of issues, but You will solve them all, eventually. Trust me!

Author

The author has long history with computers and IT systems, he proclaims to be the solution finder, the healer of the sore IT systems and acknowledged mentor. Go and find out by Yourself by following him in LinkedIn or GitHub.

https://www.linkedin.com/in/markopeltojoki/

https://github.com/MarkoPeltojoki