MASTER PROGRAMME:

«INFORMATION MANAGEMENT IN LIBRARIES, ARCHIVES, MUSEUMS»

DEPARTMENT OF ARCHIVAL, LIBRARY AND INFORMATION STUDIES

SCHOOL OF MANAGEMENT, ECONOMICS AND SOCIAL SCIENCES

Deep-Learning vs classical Machine Learning comparison for Text Classification

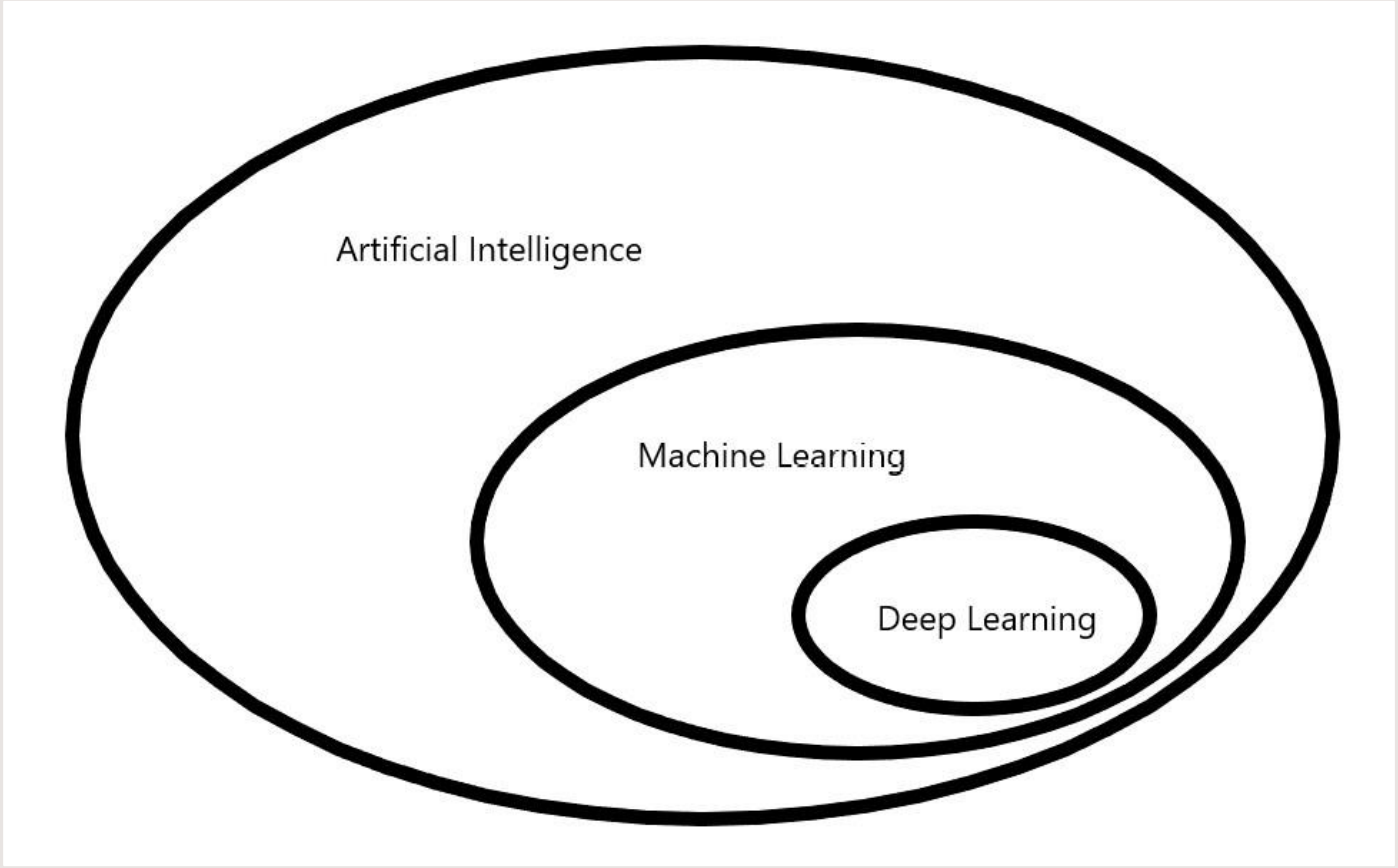
Student: Ioannis Drizis

Supervisor: Ioannis Triantafyllou

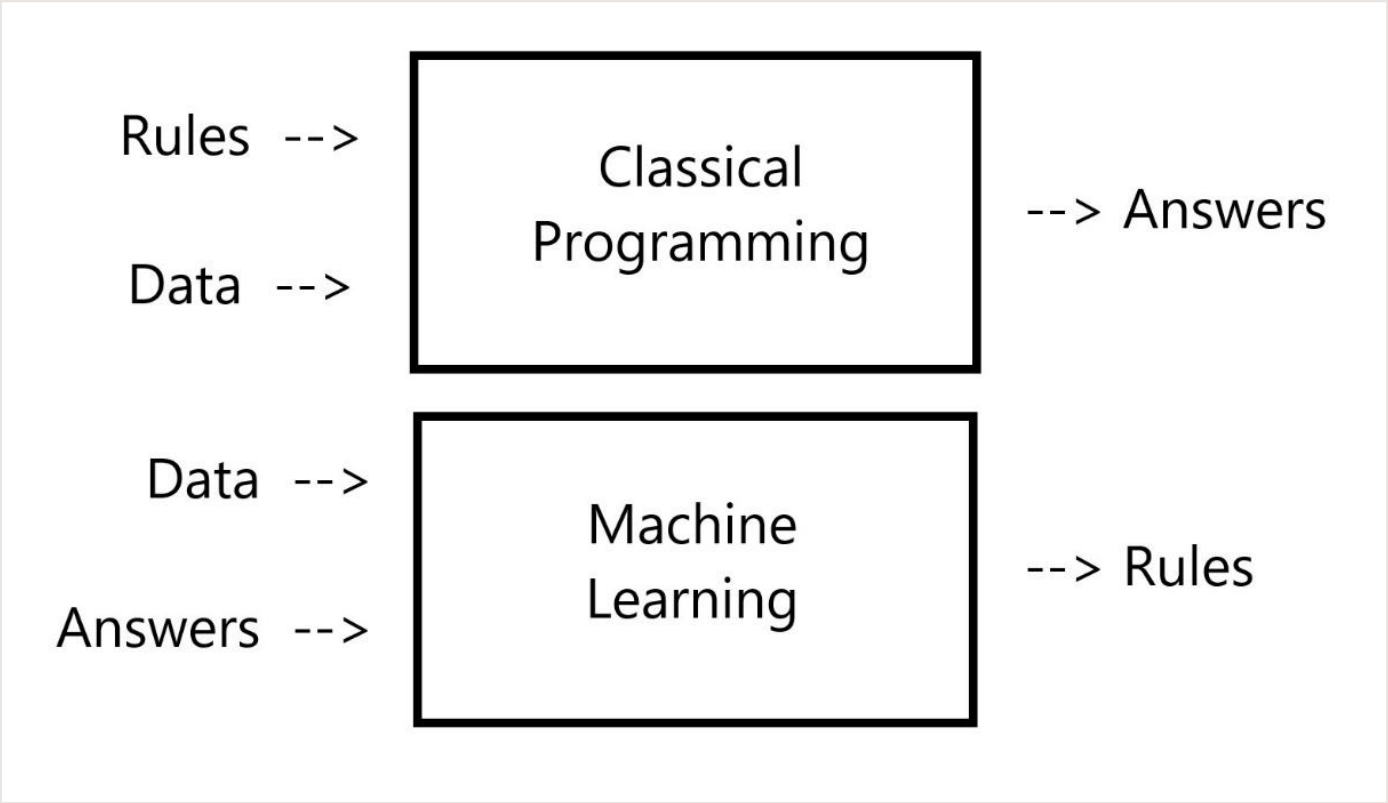
Ioannis Triantafyllou

Digitally signed by Ioannis Triantafyllou
Date: 2022.07.11 14:44:20 +03'00'

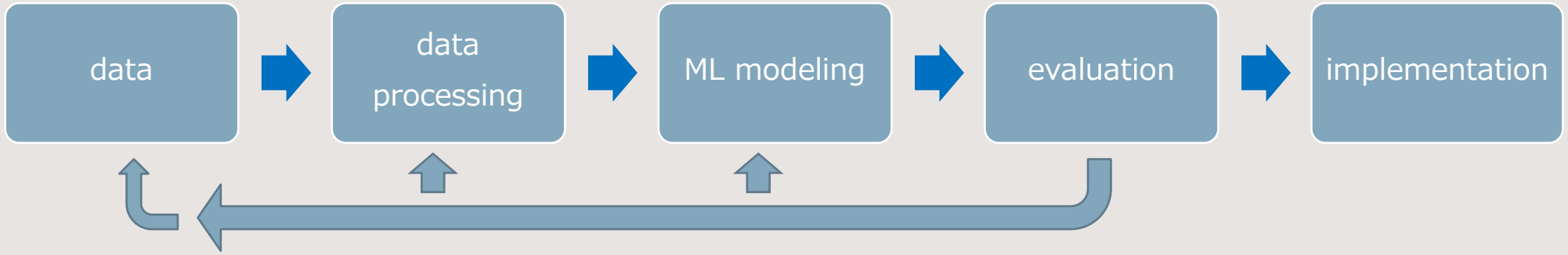
The frame of Machine Learning and Deep Learning



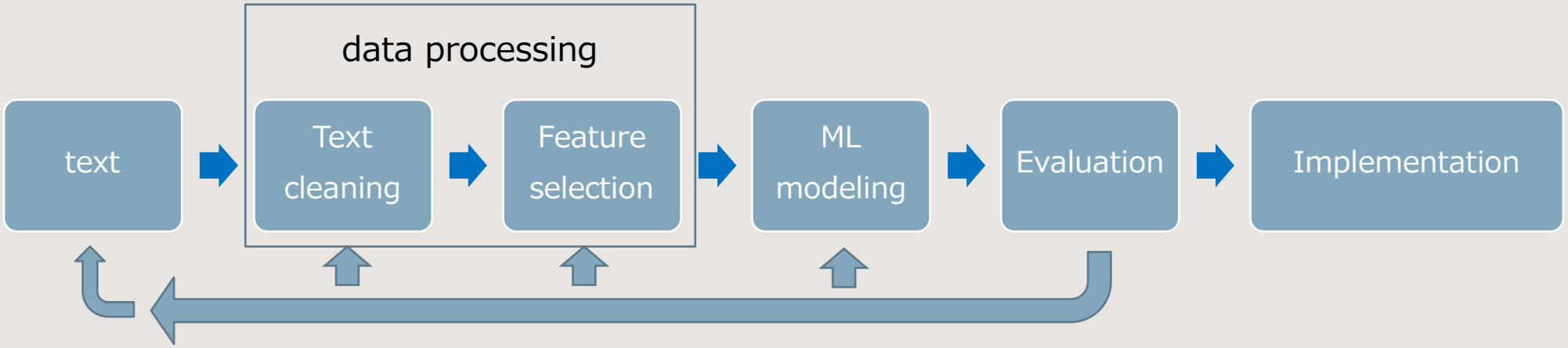
classical Programming
vs
Machine Learning



The Machine Learning pipeline in general

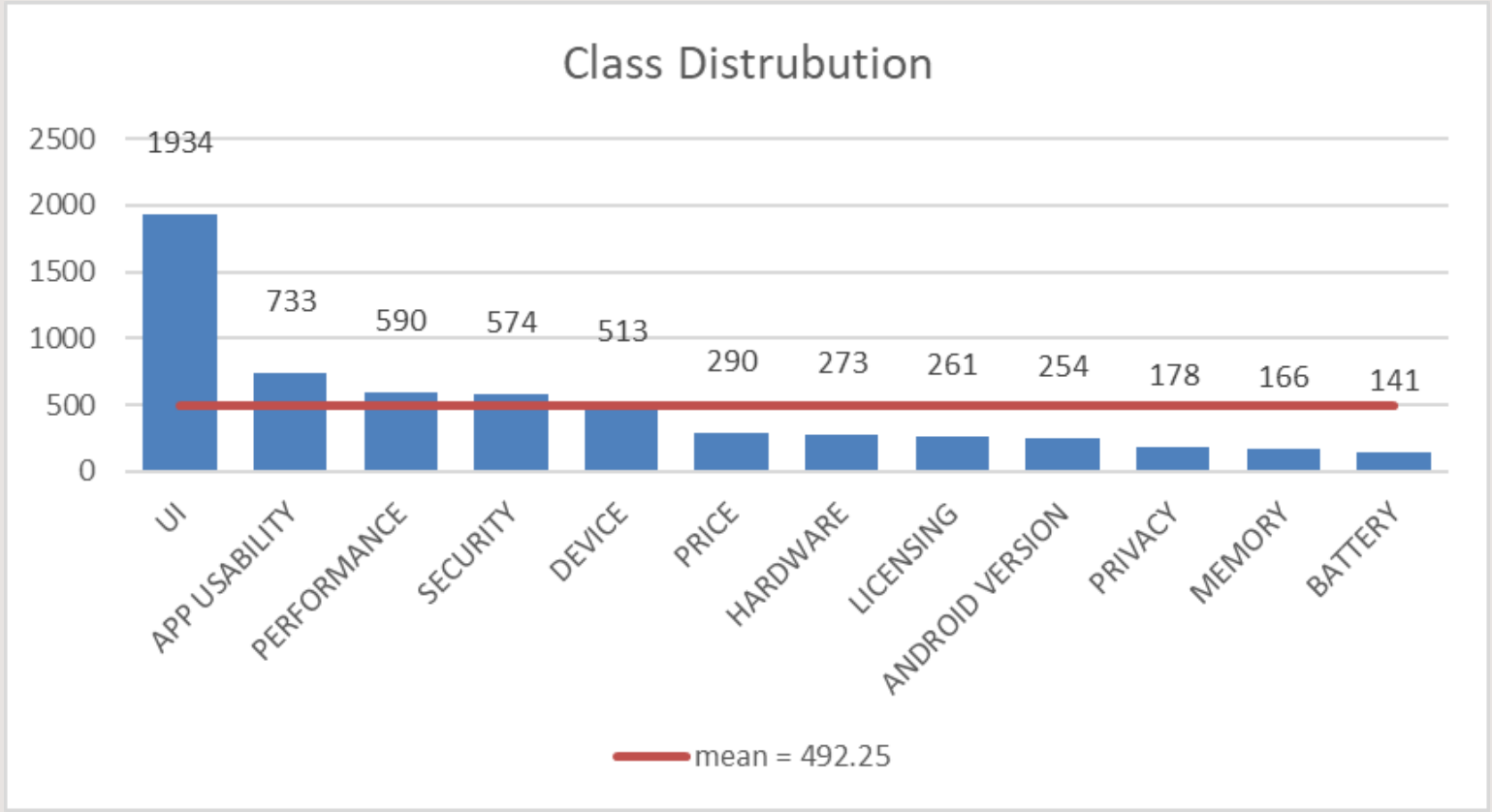


The Machine Learning pipeline for Text Classification



Dataset

- 7,754 mobile app reviews
- 12 classes
- Multi-labeled
- 3,713 → at least one class
- 4,040 → no class



Dataset: https://github.com/panichella/UserReviewReference-Replication-Package/tree/URR-v1.0

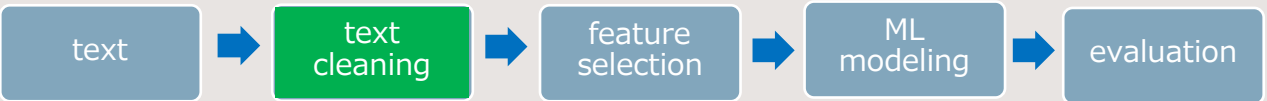
text → text cleaning → feature selection → ML modeling → evaluation

Text Cleaning

“I don’t like this application. It is too slow. Plz update!!”

[“i”, “dont”, “like”, “this”, “application”, “it”, “is”, “too”, “slow”, “plz”, “update”]

- Words in corpus: 100.820
- Lexicon = unique words in corpus: 6.963
- Grouped Lexicon = all similar words of Lexicon that are grouped: 5.559



Feature Selection

Tf.idf

Chi square

Devmax.DF

Index	Words
1	['batteri', 'battl', 'battery-sav', 'battery-typ', 'batterybot', 'battri', 'battlecri']
2	['lock']
3	['memori', 'memor', 'memoris', 'memorizey', 'memoria']
4	['ad']
5	['free']
6	['simpl', 'simplesment', 'simplic', 'simpli', 'simplest', 'simpler', 'simplist', 'simpel', 'simplier', 'simplif', 'simplifi']
7	['secur']
8	['easi']
9	['galaxi']
10	['comic', 'comicrack']

Vectorizing

["i", "dont", "like", "this", "application", "it", "is", "too", "slow", "plz", "update"]

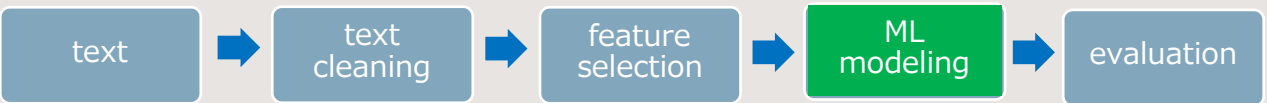
[0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, ⋯, 0, 1]

- The length of the numeric vector is equal to the size of the Lexicon that is decided to be used.
- The above numeric vector is the Bag of Words

Classical Machine Learning Models

- Naïve Bayes
- K-NN
- SVM
- Random Forest
- Logistic Regression

via Sci-Kit Python Library

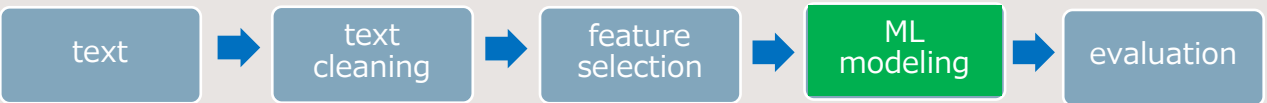


Deep Learning Models

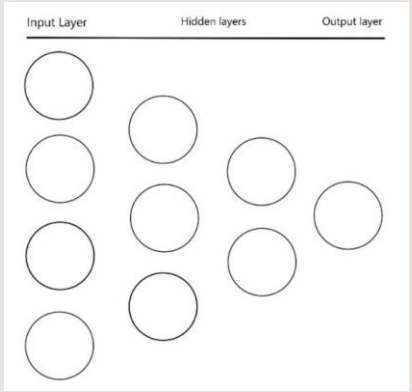
- Dense models
- CNN models
- LSTM models

via Keras Python Library

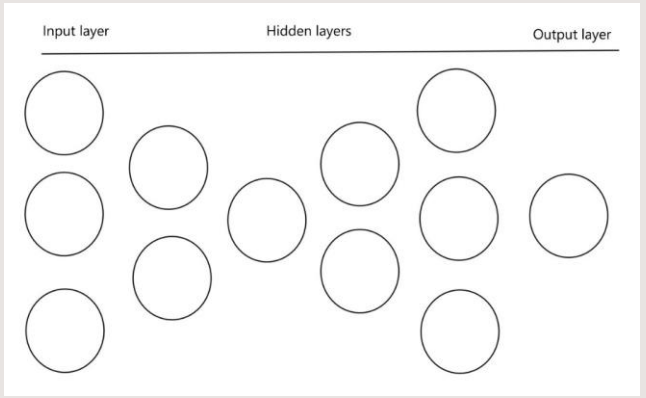
Note: Google's Colab cloud service was used due to memory shortage of own recourses. Via this service I could run Python code for DL efficiently and fast



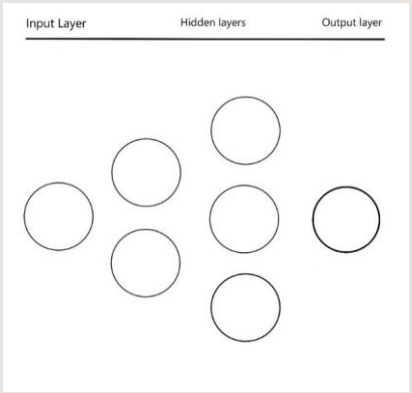
Deep Learning Models (dense models)



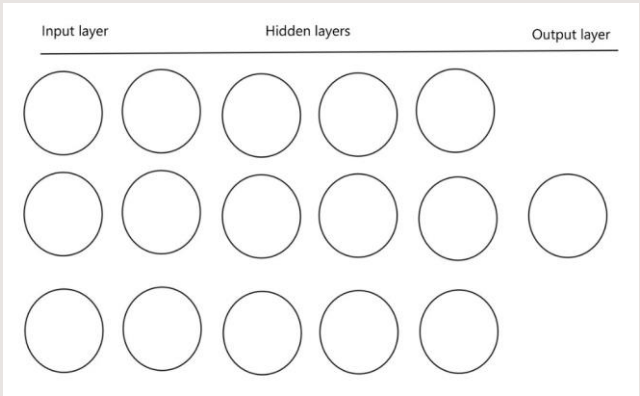
pyramid architecture



combination of pyramid and reverse pyramid architecture



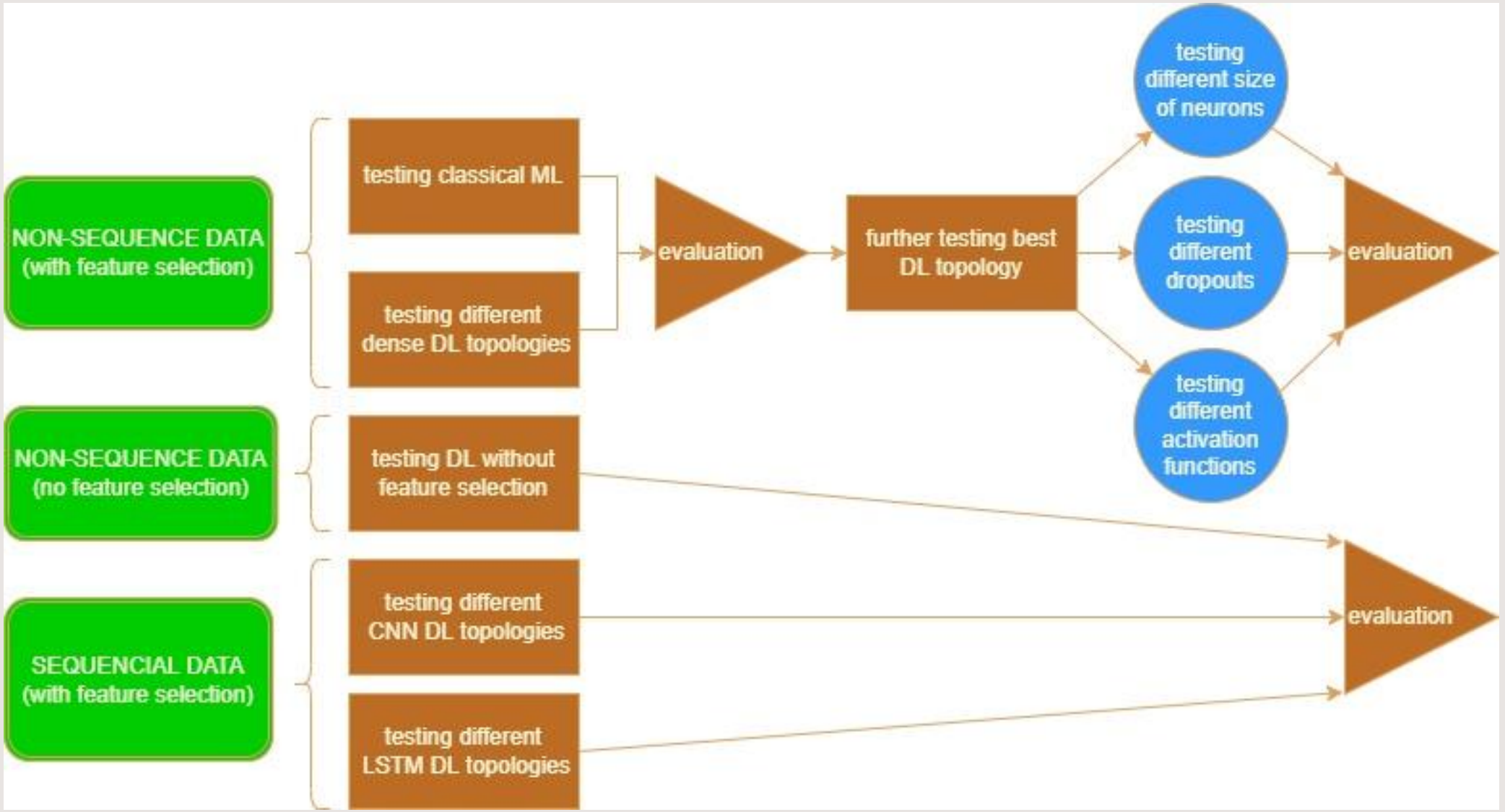
reverse pyramid architecture



Example of linear architecture

text → text cleaning → feature selection → ML modeling → evaluation

Overview of methodology

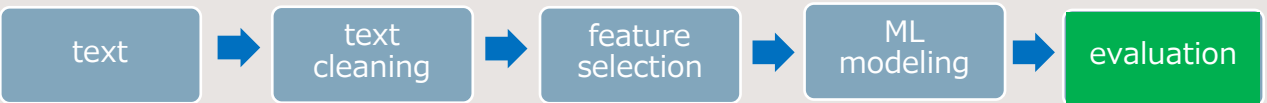


text → text cleaning → feature selection → ML modeling → evaluation

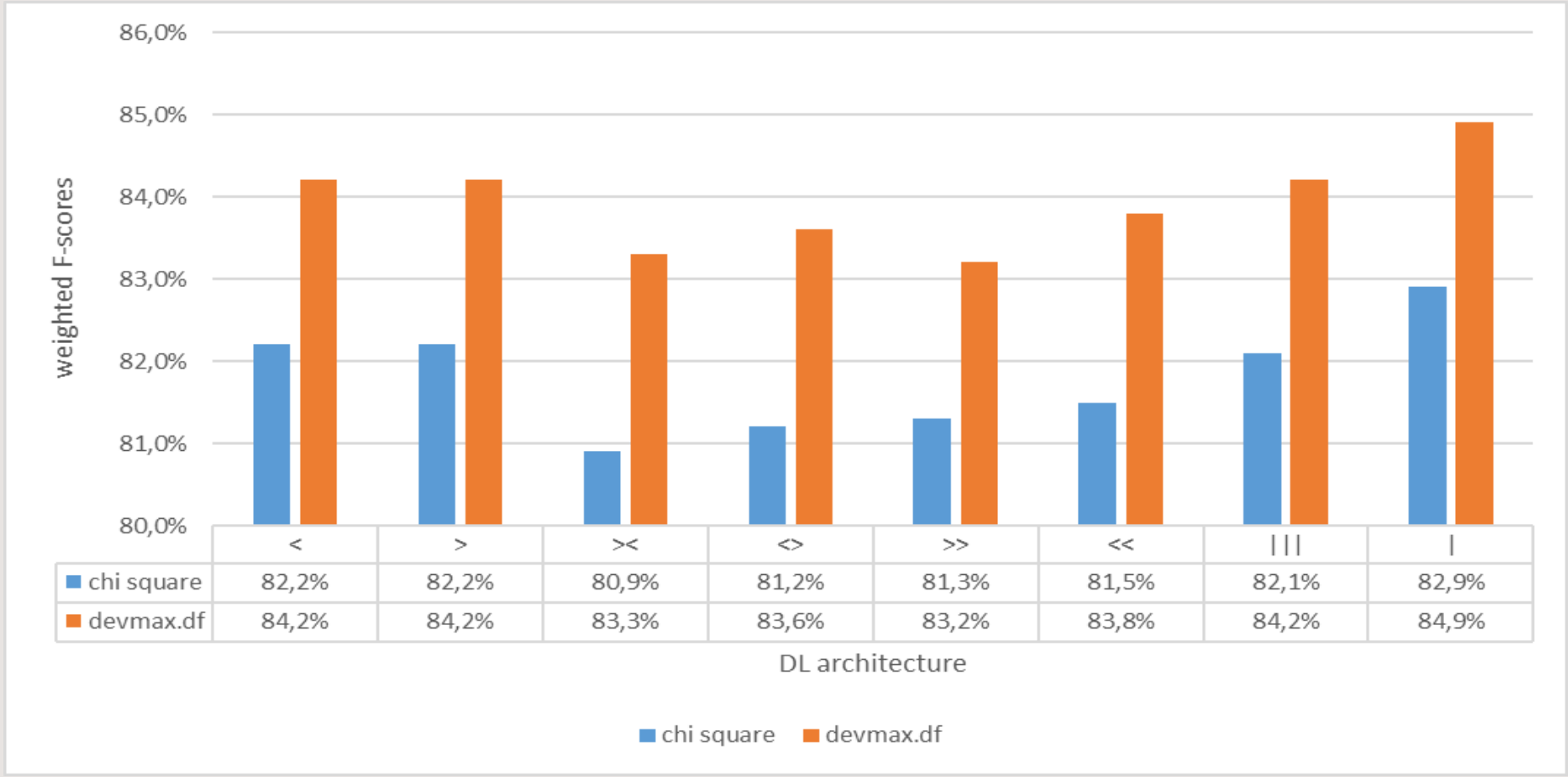
Evaluation (models created via sci-kit Learn)

Model	Feature Selection Metric	Input Vector Size	F-score
NB	x^2	200	72,8%
k-NN	devmax	100	52,5%
SVM	x^2	200	81,7%
RF	devmax	100	78,3%
LR	devmax	300	82,6%
DL (simple)	devmax	300	81,1%

Best weighted F-scores of running Classical ML models and DL via the sci-kit Python Library with different parameters



Evaluation (models created via Keras)



	<	>	><	<>	>>	<<	|||	|
chi square	82,2%	82,2%	80,9%	81,2%	81,3%	81,5%	82,1%	82,9%
devmax.df	84,2%	84,2%	83,3%	83,6%	83,2%	83,8%	84,2%	84,9%

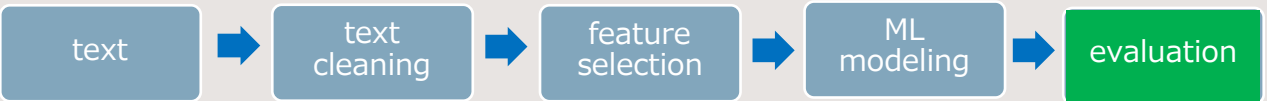
Best weighted F-scores per DL architecture by using chi square and devmax.DF feature extraction metric

text → text cleaning → feature selection → ML modeling → evaluation

Evaluation (models created via Keras)

Tests with Deep Learning by using only one hidden layer and by using different hidden layer's unit sizes and different dropout

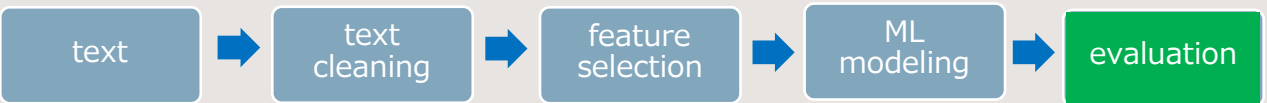
		Dropout Chance								
		10%	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**
multiplier	**1**	84,7%	85,0%	84,7%	84,6%	84,7%	84,5%	84,1%	83,2%	76,1%
	2	84,7%	85,0%	84,8%	84,8%	84,8%	84,7%	84,7%	84,2%	82,8%
	3	85,0%	84,8%	84,8%	84,9%	84,7%	**85,1%**	84,8%	84,6%	83,8%
	…	…	…	…	…	…	…	…	…	…
	39	84,8%	84,7%	84,9%	84,8%	84,9%	84,9%	84,9%	84,8%	85,0%
	40	84,6%	84,8%	84,8%	84,8%	84,7%	84,8%	84,9%	85,0%	84,9%
	41	84,6%	84,7%	84,8%	84,9%	84,8%	84,7%	84,9%	84,9%	85,1%
	42	84,7%	84,5%	84,6%	84,9%	84,9%	84,8%	85,0%	84,9%	85,1%
	43	84,7%	84,7%	84,8%	84,8%	84,8%	84,8%	85,0%	85,0%	85,1%
	44	84,6%	84,7%	84,8%	84,7%	84,8%	84,7%	85,0%	84,9%	85,1%
	45	84,6%	84,5%	84,8%	84,9%	84,8%	84,9%	84,8%	85,0%	**85,2%**
	46	84,8%	84,6%	84,6%	84,8%	84,9%	84,9%	84,9%	84,9%	85,0%
	47	84,7%	84,8%	84,6%	84,8%	84,9%	84,7%	84,9%	85,0%	85,1%
	48	84,6%	84,4%	84,9%	84,8%	84,7%	84,9%	84,8%	85,1%	85,0%
	49	84,7%	84,5%	84,8%	84,6%	85,0%	84,8%	84,8%	85,0%	85,1%
	50	84,5%	84,7%	84,5%	84,5%	84,8%	84,6%	84,7%	84,6%	84,6%



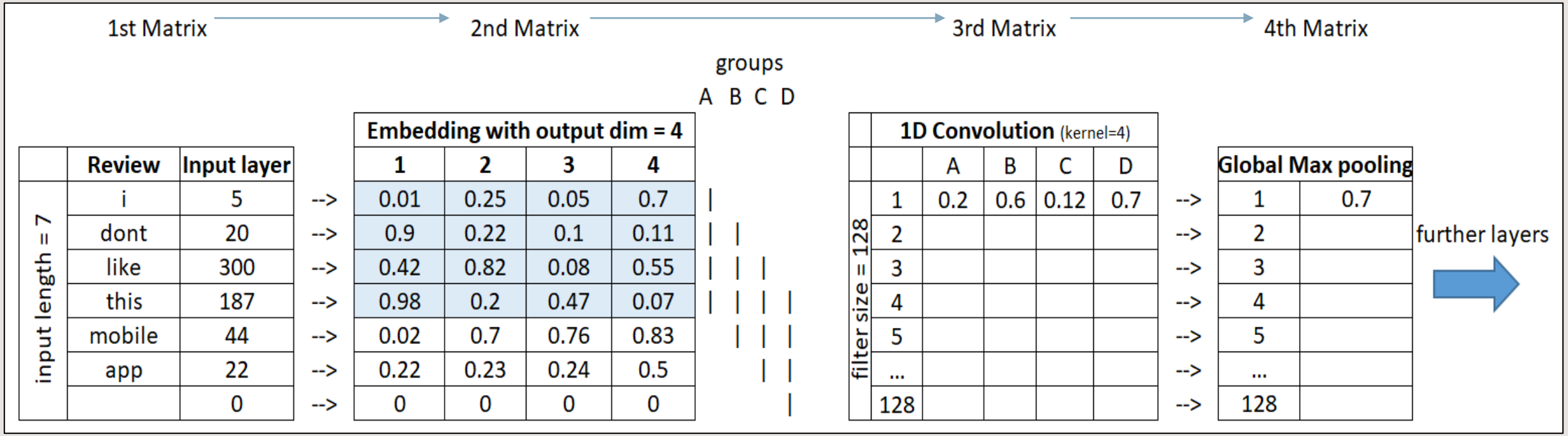
Evaluation (models created via Keras)

Activation function	F-score
relu	83,9%
selu	84,5%
sigmoid	75,6%
softplus	77,5%
softsign	84,4%
tanh	84,4%
elu	84,5%
exponential	77,4%

Performance of testing different activation functions with a one-hidden layer DL NN



Deep Learning Models (CNN network)



	Review	Input layer		1	2	3	4
input length = 7	i	5	-->	0.01	0.25	0.05	0.7
	dont	20	-->	0.9	0.22	0.1	0.11
	like	300	-->	0.42	0.82	0.08	0.55
	this	187	-->	0.98	0.2	0.47	0.07
	mobile	44	-->	0.02	0.7	0.76	0.83
	app	22	-->	0.22	0.23	0.24	0.5
		0	-->	0	0	0	0

		A	B	C	D			
filter size = 128	1	0.2	0.6	0.12	0.7	-->	1	0.7
	2					-->	2	
	3					-->	3	
	4					-->	4	
	5					-->	5	
	...					-->	...	
	128					-->	128	

Demonstration example of a CNN network

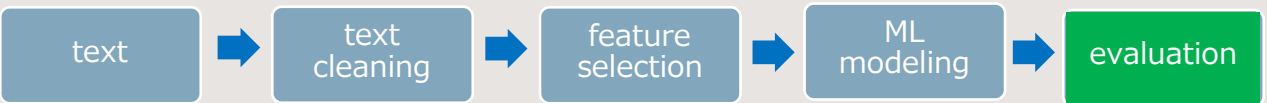
Evaluation (models created via Keras)

Dense layer without feature selection	
Multiplier	**F-score**
1	79,9%
2	80,0%
3	80,4%
4	80,1%
5	80,0%

CNN	
Multiplier	**F-score**
1	83,5%
2	83,7%
3	83,8%
4	83,7%
5	83,5%

LSTM	
Multiplier	**F-score**
1	74,9%
2	77,4%
3	77,4%
4	78,1%
5	76,7%

Performance of testing different DL architecture topologies with different parameters



Conclusions

- A tuned DL model performed better than other ML models
- A “shallow" DL architecture topology produced better results than more “deep” ones
- The feature selection metric Devmax.DF outperformed tf.idf and chi square
- Results cannot be generalized. More research is needed with more datasets, and with more in-depth analysis of CNN and LSTM models
- Overfitting may have occurred. This needs further research to validate models’ generalization capabilities

Model	Feature selection metric	Input Vector Size	F-score
NB	x^2	200	72,8%
k-NN	Devmax.DF	100	52,5%
SVM	x^2	200	81,7%
RF	Devmax.DF	100	78,3%
LR	Devmax.DF	300	82,6%
DL (simple)	Devmax.DF	300	81,1%
DL	Devmax.DF	300	85,2%

Conclusions - Interpretation

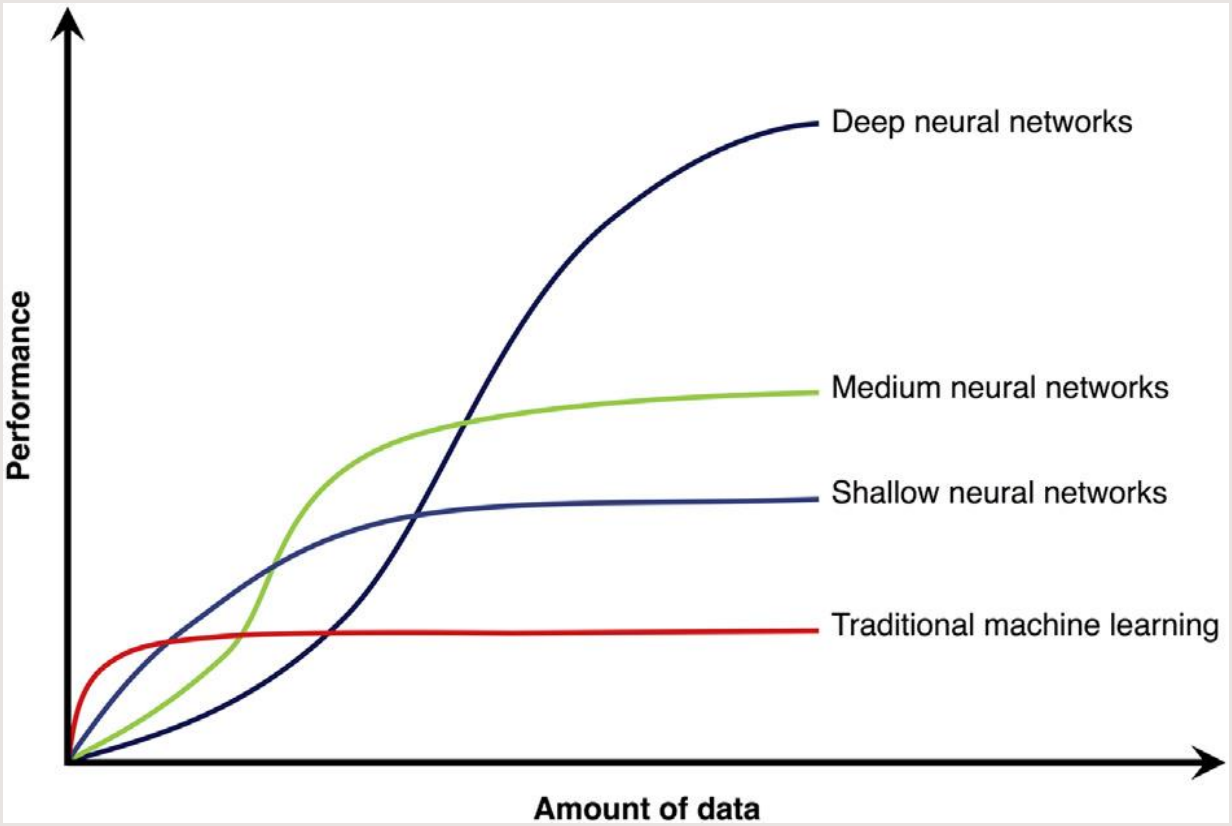


Figure received from paper “Canadian Association of Radiologists White Paper on Artificial Intelligence in Radiology” (2018), reference within paper:
Bahnsen AC. Easy Solutions, Inc. Building AI applications using deep learning. Available at: http://blog.easysol.net/building-ai-applications. Accessed January 28, 2018.

Thank you for your attention!

Questions?!

Student: Ioannis Drizis

Supervisor professor: Ioannis Triantafyllou