Combining cancer research, next generation sequencing and Machine Learning

(so-called Buzzword galore)

-The art of applying machine learning to data that is not really fit for it

Ceren Kocak, Christoffer S. Bangsgaard,
Lau K. Vestergaard & Signe Poulsen

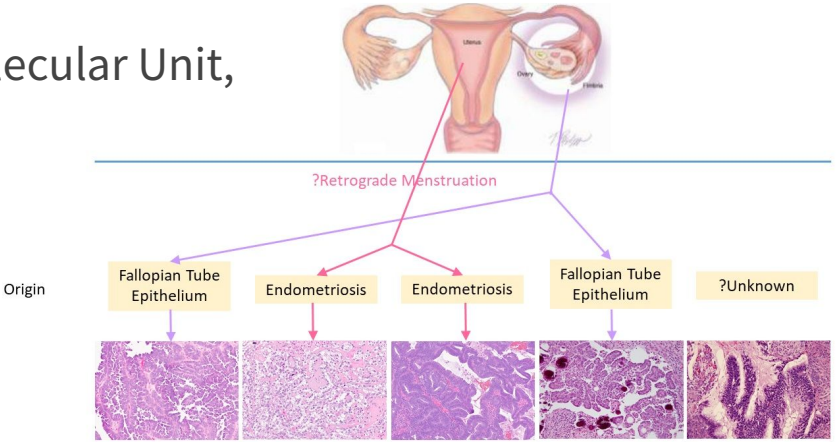
- All members of the group contributed equally to the project.

Outline of presentation

- Brief intro
- Understanding the data (and the preprocessing)
- Aim
- Working with data per mutation
  - Dimensionality reduction
  - Clustering
- Working with data per patient
  - PCA
  - Oversampling and auto encoding
  - Classification
- Evaluating results
- Overall conclusion
- Appendix

Brief introduction to ovarian cancer

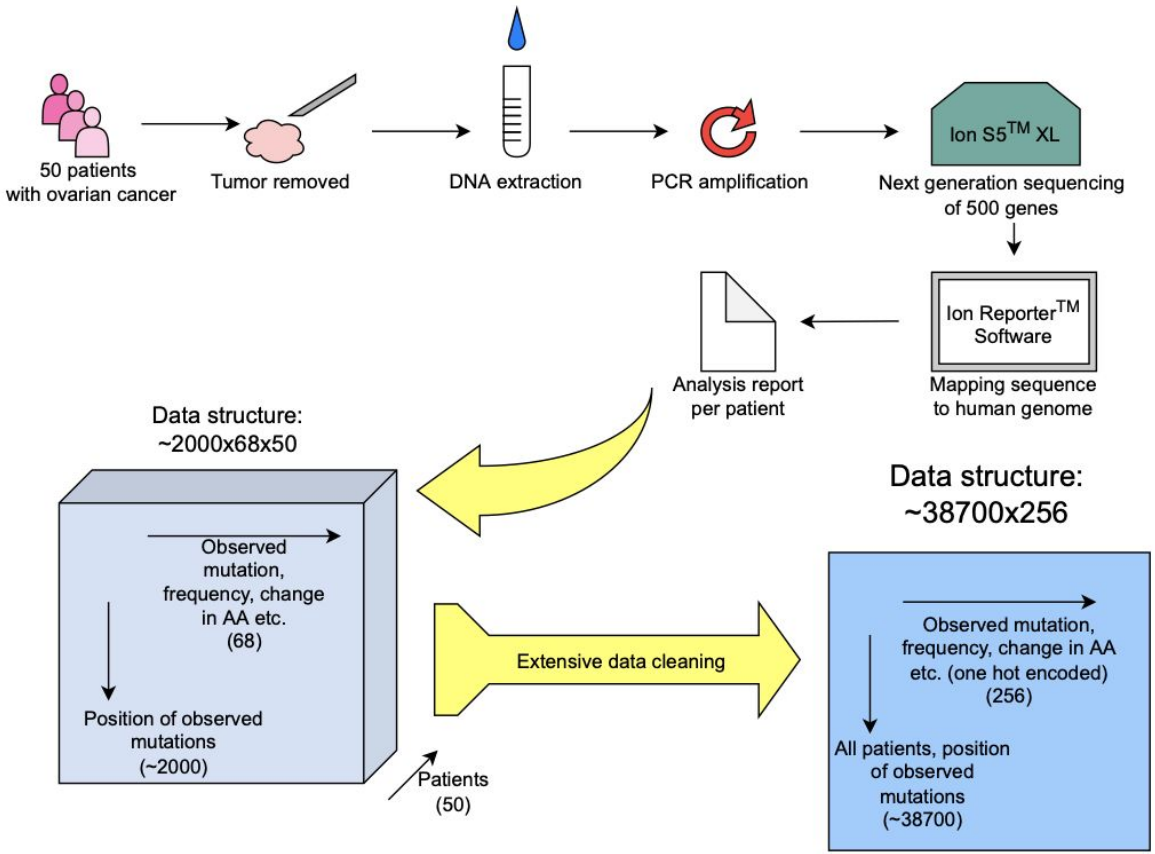
- Ovarian Cancer ranks 5th in cancer deaths among women and is the most lethal gynecologic malignancy.
- Poor 5 year survival rate in late stages
- There are 5 subtypes of ovarian cancer
- Typed using microscopy
- Data was kindly made available by the Molecular Unit, Department of Pathology, Herlev Hospital.



	High-Grade Serous Carcinoma	Clear Cell Carcinoma	Endometrioid Carcinoma	Low-Grade Serous Carcinoma	Mucinous Carcinoma
% of all Ovarian Carcinomas	~70%	~10%	~10%	<5%	<5%

Adapted from: http://www.bccancer.bc.ca/books/ovary-epithelial-carcinoma/histological-classification-of-ovarian-carcinoma

The data



Patient domain:

Cancer subtype representation:

High Grade Serous Carcinoma:	38 (76%)
Ovarian Clear Cell Carcinoma:	4 (8%)
Mucinous Carcinoma:	4 (8%)
Endometrioid Carcinoma:	3 (6%)
Fallopian Tube Carcinoma:	1 (2%)

Cancer stage representation:

I	II	III	IV
8 (16%)	5(10%)	30 (60%)	7 (14%)

Data structure

Mutation domain: ~38.000

Location of mutation

Observed mutation

Type of mutation

Gene

Type of location

% frequency of observed mutation

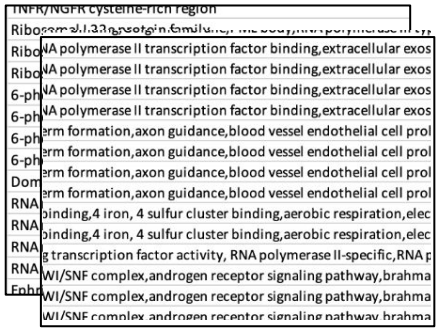
AA change caused by mutation

~38.000

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	locus	genotype	filter	ref	served_alle	type	subtype	no_call_reason	cnv_p-value	genes	location	length	hine_varian	mine_gene	opy_numbe	cytoband	info	variant_id	ariant_nam	%_frequency	amino_acid_change
2	chr1:2491	T/T	PASS	C	T	SNV				TNFRSF14	TNFRSF14:intronic:NM_003820.3	1						vc.novel.3		99,59	p.?
3	chr1:2491	C/T	PASS	C	T	SNV				TNFRSF14	TNFRSF14:exonic:NM_003820.3	1						vc.novel.4		2,69	p.Arg109Trp
4	chr1:6252	CCACAC/CC	PASS	CCACAC	CCAC	INDEL				RPL22	RPL22:intronic:NM_000983.4	2						vc.novel.5		45,97	p.?
5	chr1:6257	T/C	PASS	T	C	SNV				RPL22	RPL22:intronic:NM_000983.4	1						vc.novel.6		49,87	p.?
6	chr1:6257	G/A	PASS	G	A	SNV				RPL22	RPL22:intronic:NM_000983.4	1						vc.novel.7		49,15	p.?
7	chr1:1045	G/A	PASS	G	A	SNV				PGD	PGD:intronic:NM_002631.4	1						vc.novel.9		53,85	p.?
8	chr1:1046	T/C	PASS	T	C	SNV				PGD	PGD:exonic:NM_002631.4	1						vc.novel.10		51,8	p.Asp40=
9	chr1:1047	C/T	PASS	C	T	SNV				PGD	PGD:exonic:NM_002631.4	1						vc.novel.11		51,15	p.Asp244=
10	chr1:1047	G/A	PASS	G	A	SNV				PGD	PGD:intronic:NM_002631.4	1						vc.novel.12		46,53	p.?

Furthermore:

- Protein family (words)
- Gene ontology (words)
- Verdict (Pathogenic, benign, unknown)



Data cleaning by:

- Removing columns that contain redundant or limited information
- One hot encode (e.g. Amino acids)
- Term frequency (TF-IDF) to find 100 most frequent words for protein family and gene ontology
- Converting character data to numerical ordinal

Aim

Overall aim

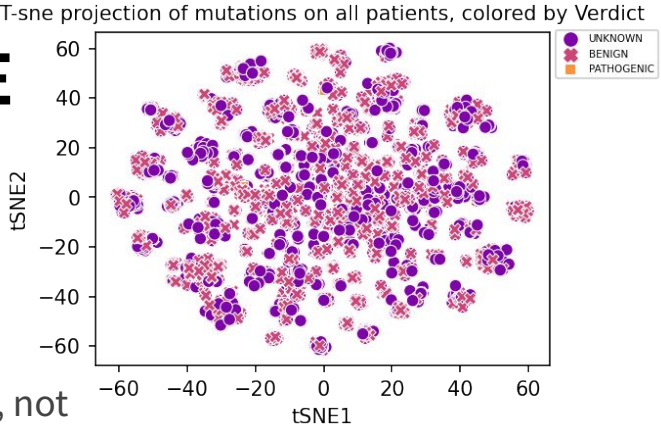
- To explore what these large datasets tell us about each patient.
- To investigate the possibility of extracting data on a mutation and patient basis
- Knowing more about patient mutation profiles may help doctors prioritize treatment options for the individual patient

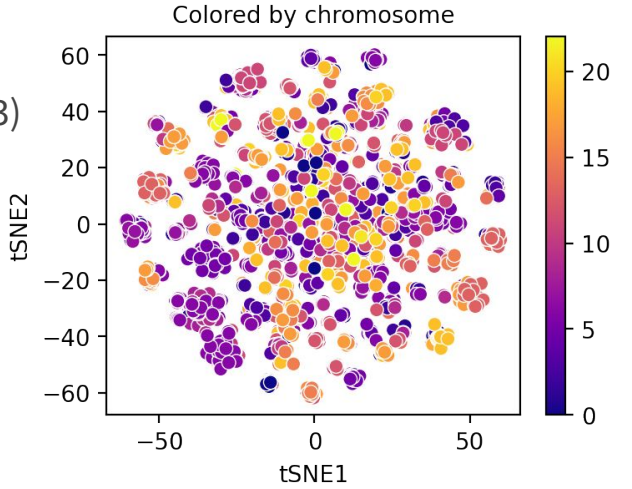
How will this be obtained?

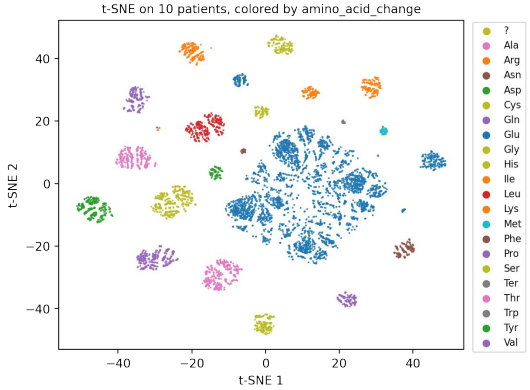
- Clustering (unsupervised) → Obtain information concerning Unknown mutations.
- Do these group close to Benign mutations or Pathogenic mutations?
  - t-distributed stochastic neighbor embedding (t-SNE)
  - Clustering
- Classification (supervised) → Prediction of cancer subtype
  - Oversampling and Auto Encoding
  - Convolutional Neural Network (CNN)

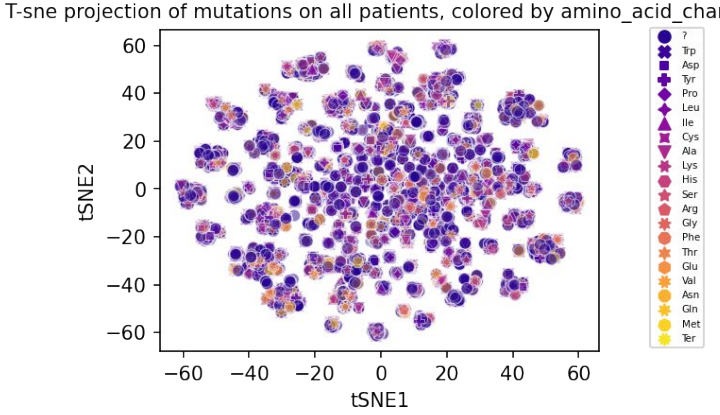
Reducing dimensionality with t-SNE

- Look at data in the mutation domain
- t-SNE to two dimensions, color by different features to investigate the clustering
- Color by all features, some information but not much
- Hyperparameter optimization - perplexity
- At first on 10 patients, nice separation, then applying to all 50 patients, not as clear picture
- Color by verdict, no clear picture about the unknown mutations
- Mutation distribution: Unknown (15.866), Benign (22.844), Pathogenic (83)



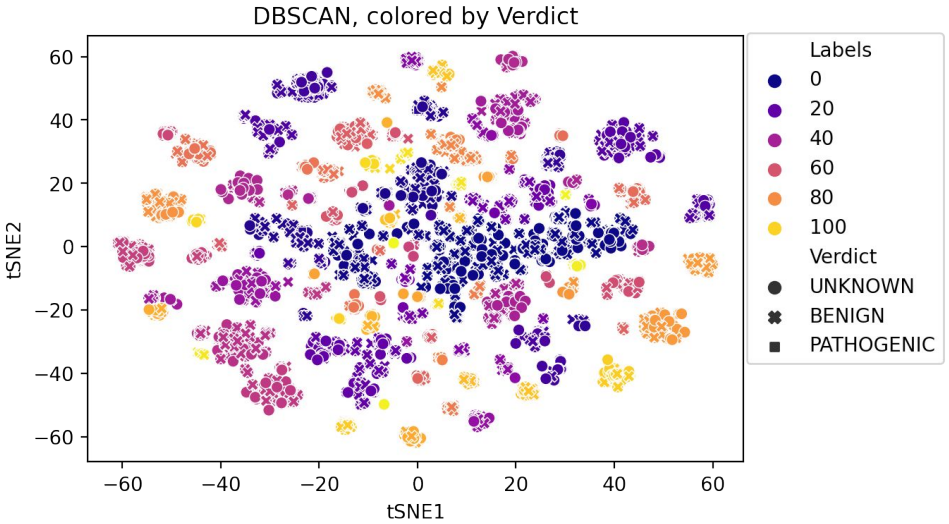


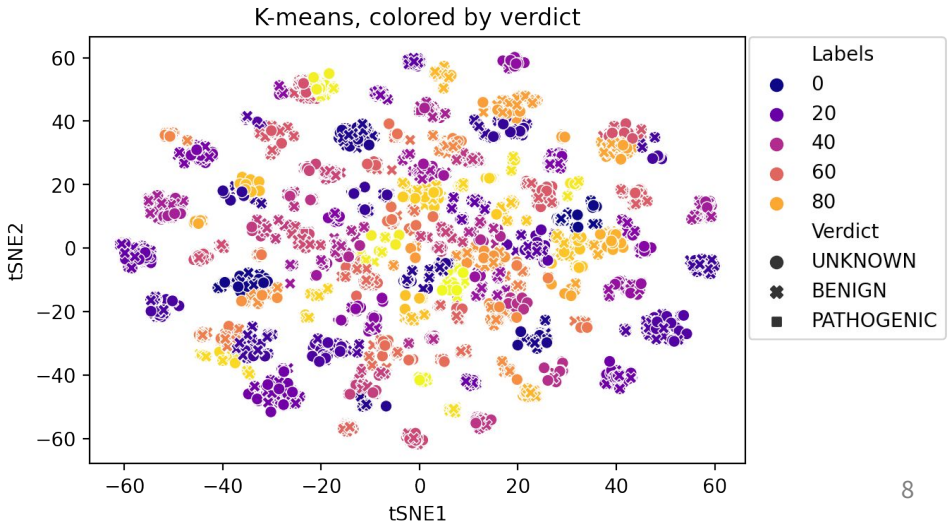




Clustering based on t-SNE

- Input data: 2 dimensions from t-SNE into DBSCAN and K-means
- Colored by cluster number
- DBSCAN and K-means perform a bit different, more clusters in DBSCAN, bigger one in the middle
- Once again illustrates that there are different trends in data, DBSCAN shows big lump of similar mutations in the middle
- In order to fully extract meaning of t-SNE and clustering more hyperparameter optimization is needed





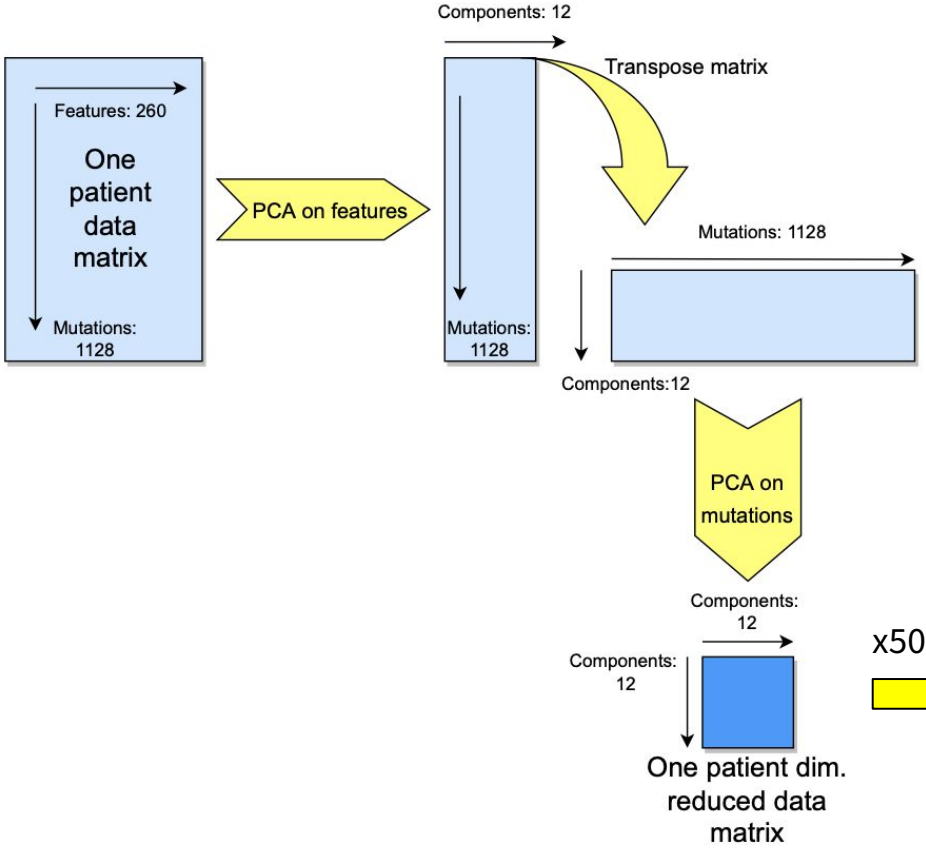
Moving on to classification

- The aim was to classify patients within subtypes of cancer (Multiclass and Binary)

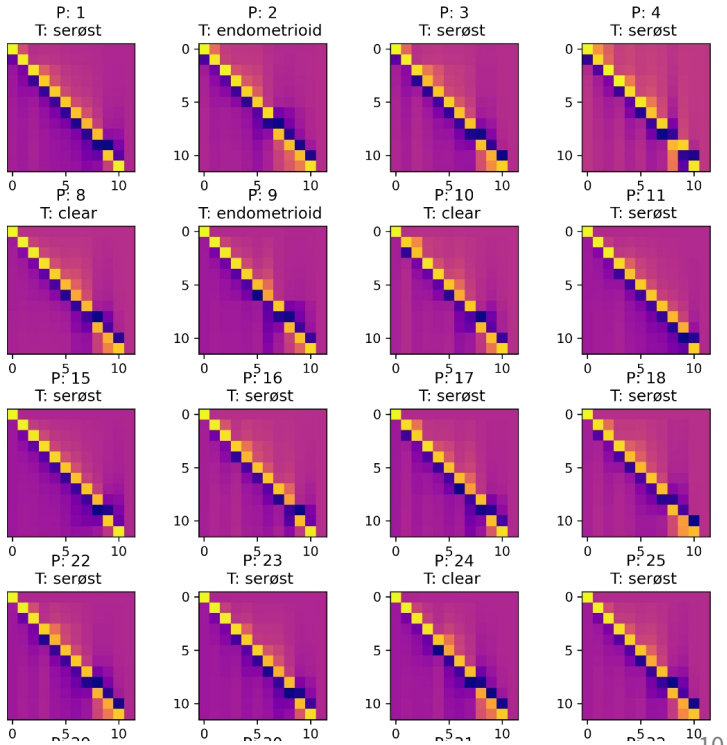
Problems to handle before classification

- Not all patients had same amount of mutations, therefore not the same length
- The classes of cancer subtypes were highly unbalanced
- We needed more patients to train on

Dimensionality Reduction with PCA



x50 patients



Generating new “patients”

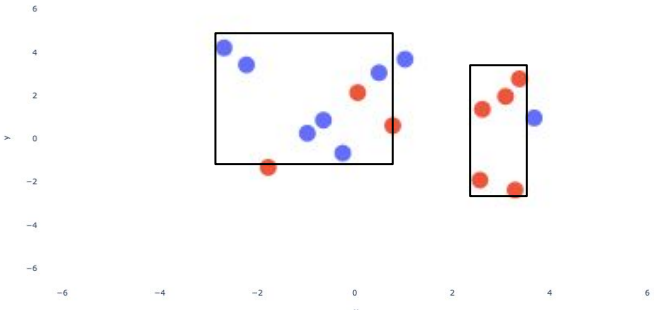
Autoencoding

- Use Variational Autoencoder to generate more patients
- Variational Autoencoder was first made for MNIST.
- Encoder: Reducing input X to a lower (gaussian dist.) dimensional space (Z)
- Decoder: uses input (Z) together with probability distribution to output.
- Inspecting latent space for multi-classification, no way to separate groups entirely, therefore decide not to perform multi-classification.

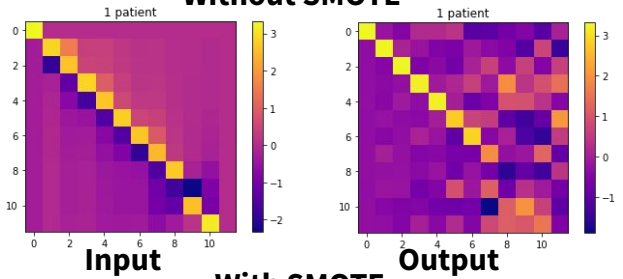
Oversampling with SMOTE

- Not satisfied with the first generated patients, decided to oversample class “other” with SMOTE before VAE.
- SMOTE visibly improved the simulated patient

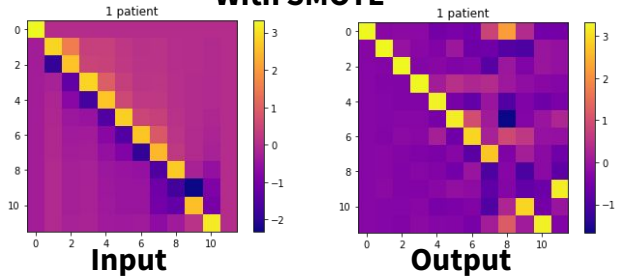
Patient representation in the 2D latent space (binary classes)



Without SMOTE

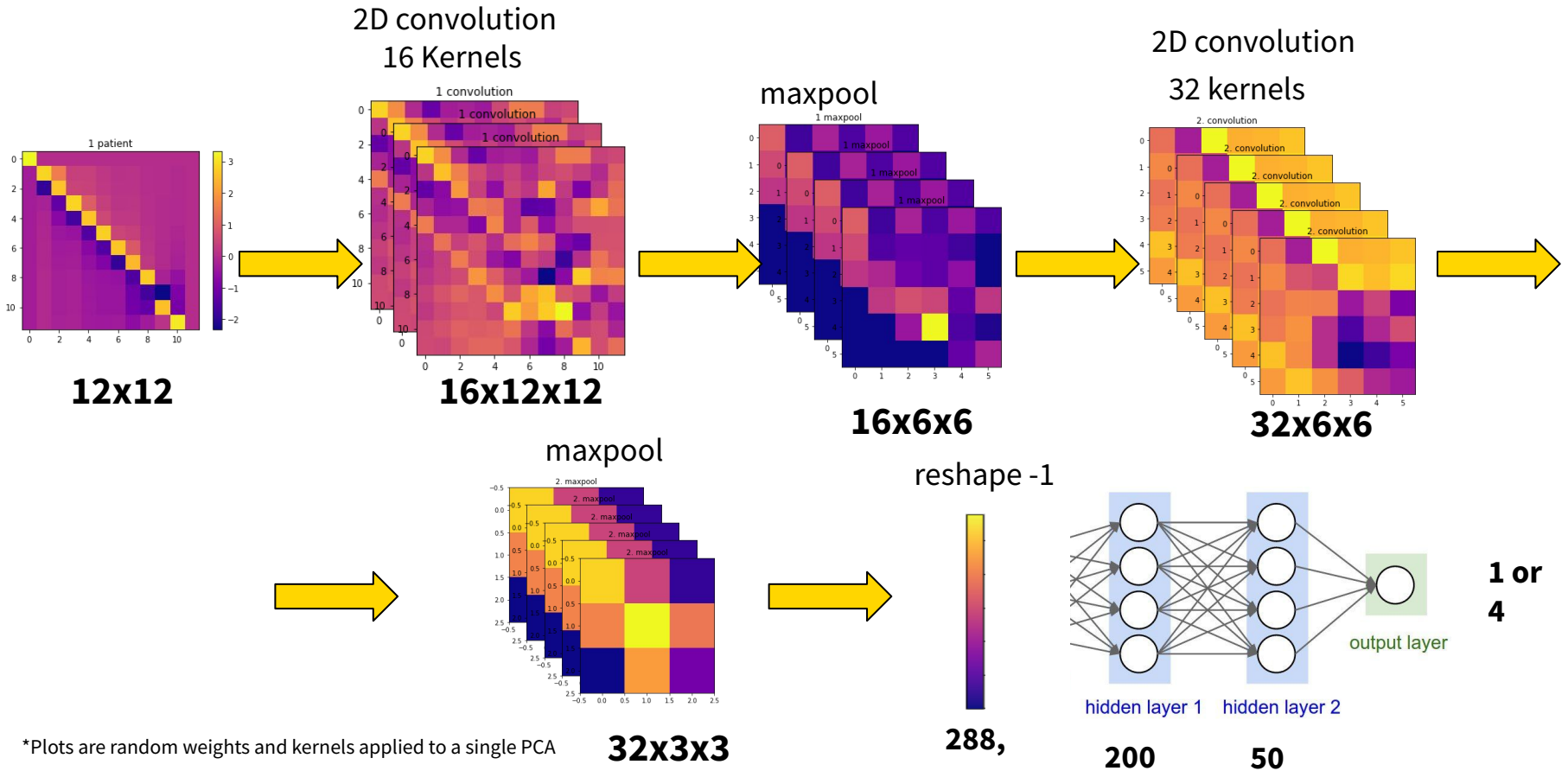


With SMOTE



Architecture of the CNN

Architecture converted from MNIST CNN



*Plots are random weights and kernels applied to a single PCA

How did we perform in classification?

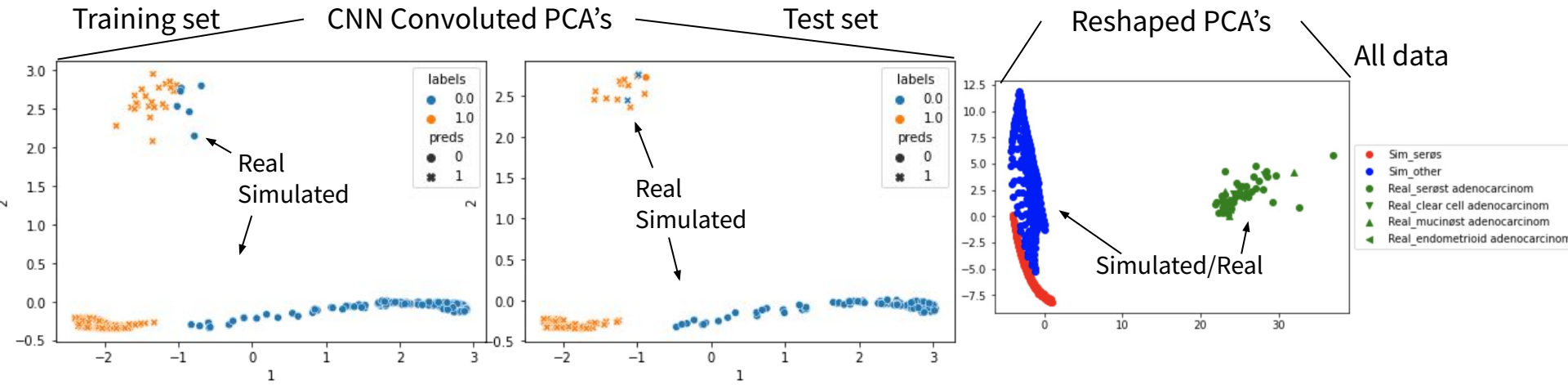
- Chose to benchmark CNN performance with boosted decision tree (LightGBM)
- Classification only on the real patients (train = 40, test = 9)
  - Results sound great right … but inspecting predictions, it predicts all samples as 1 (High Grade Serous)
- Classification on both simulated and real patients, something seems weird.
- Performs better on the simulated than on the real patients, a bit concerning

Accuracy scores (test set)	Training and test data: Real patients (49)		Training data: Both simulated and real patients Binary classification (697)		
	Binary (38+11)	Multiclass (38+4+4+3)	Only sim. patients in test-set	Both sim. and real patients in test-set	Only real patients in test-set
LightGBM	77,78%	77,78%	100%	98,56%	85,00%
CNN	80%	66,66%	100%	96,00%	66,67%

Why did the classification perform like this?

Plotting CNN convoluted PCAs and reshaped input PCAs (for LGBM)

- Real patients are separated (different) from simulated patients for both CNN and LGBM.
- The input PCA's different subtypes are closely clustered.
- CNN shows that the differences are learnable - we just don't have enough data.
  - The convoluted real patients are separated by subtype in the training set not in test set



Labels show the true cancer subtype, the shapes show the predicted subtypes(1 is Serous, 0 is other.)

Overall conclusion

Unsupervised clustering

- No clear picture for unknown-effect-mutations but most trend with Benign.
- Maybe too much information in the dataset to be mapped to 2 dimensions
- t-SNE showed potential for 10 patients, more time might have made it possible to optimize for 50 patients

Supervised classification

- ML can be done on DNA sequencing, but we need more patients to train on
- The simulated patients turned out to be too different from the real patients
    - Not good to train on, the classification of real patients were then not good.
- Even though many ML tricks were applied it is the lacking of real patients that makes the classification performance poor.

What to investigate next?

- Find more patients
- Sequencing-data is covered by GDPR legislation, hard to get access to more data/patients publicly in databases
- Divide the real patients into smaller chunks (e.g. chromosomes) and consider them as patients themselves
- Survivability predictor / how responsive a patient may be to a treatment - we have a lot of information for each patient - this information may be hidden within.

Appendix

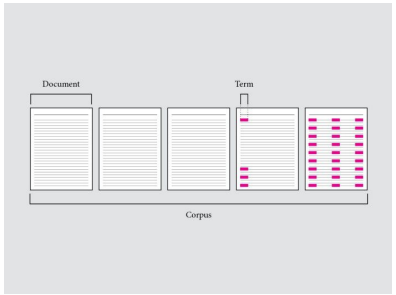
Table of contents:

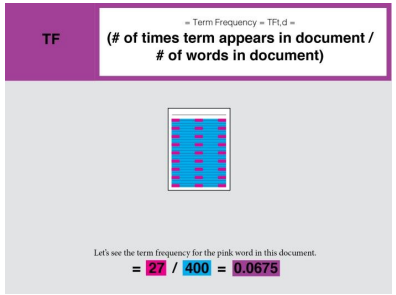
Appendix: Data Preprocessing

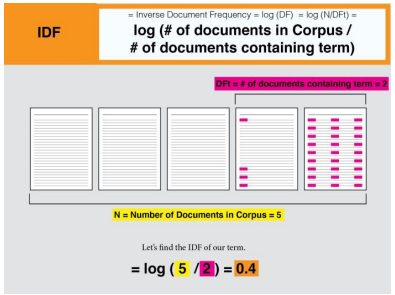
Protein family and Gene-ontology (text columns)

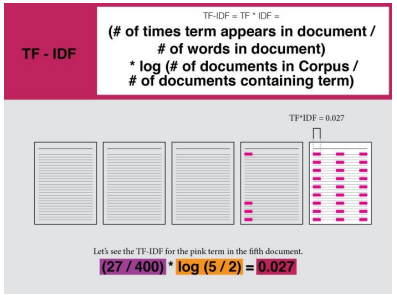
Term Frequency - Inverse Document Frequency (TF-IDF)

How the word cells were transformed into vectors

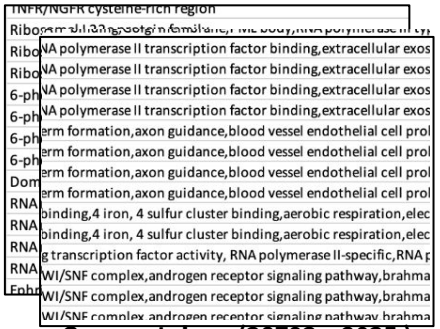




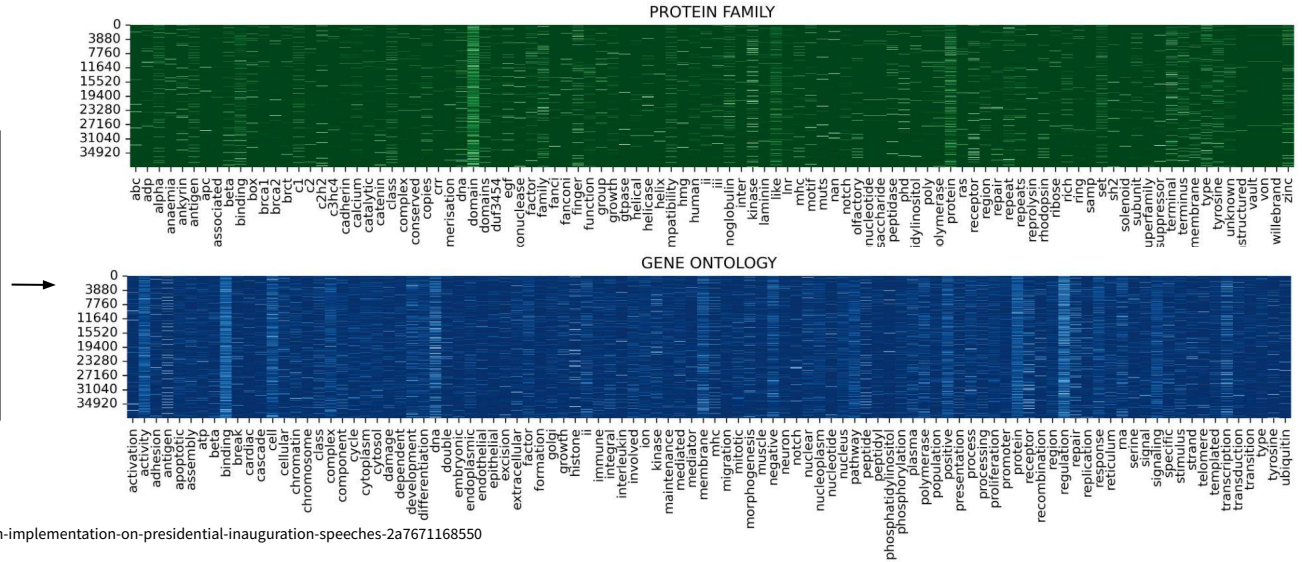




Protein family (38793 x 738)



Gene ontology (38793 x 2035)

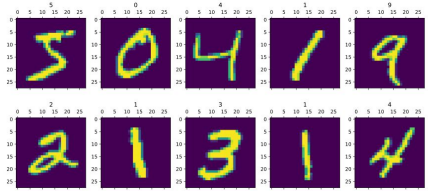


https://towardsdatascience.com/tf-idf-a-visual-explainer-and-python-implementation-on-presidential-inauguration-speeches-2a7671168550

Appendix: Simulating new patients

Variational Auto Encoder on MNIST

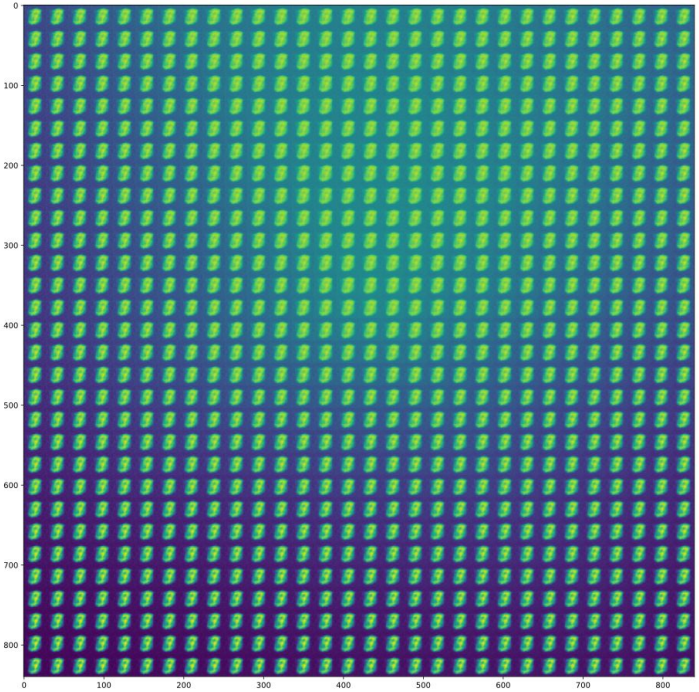
Input



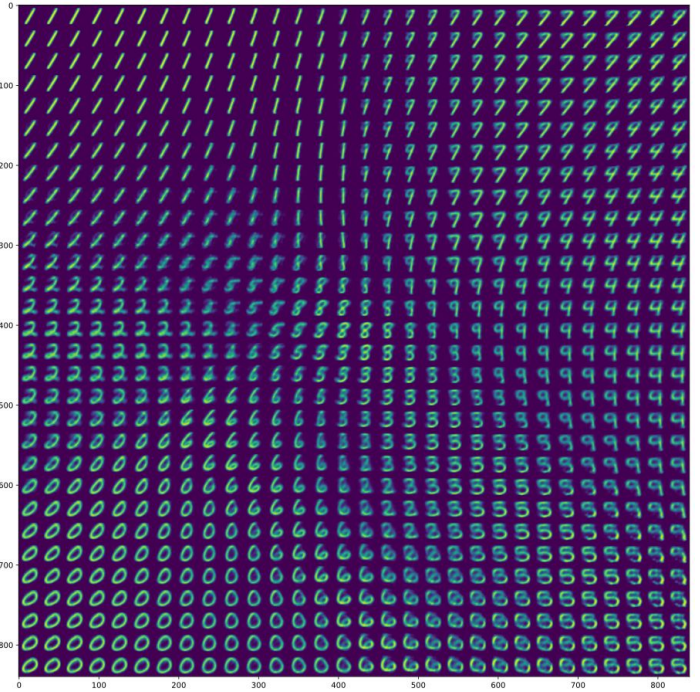
We tried to applicate our situation of 76 patient to the MNIST data set, which point towards that our issue is highly related to data size.

It is clear to see that the output of the VAE performs much worse when input is only 76 cases

Dataset size: 76



Dataset size: 70000

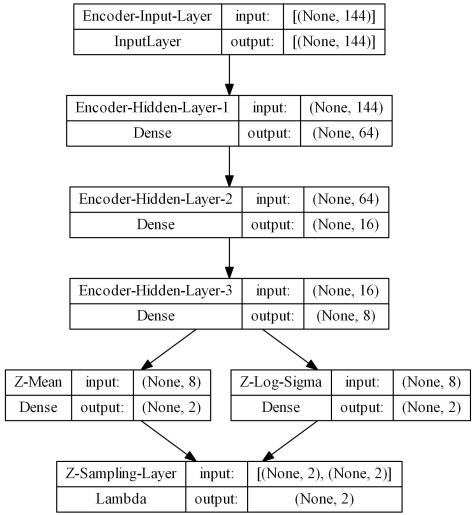


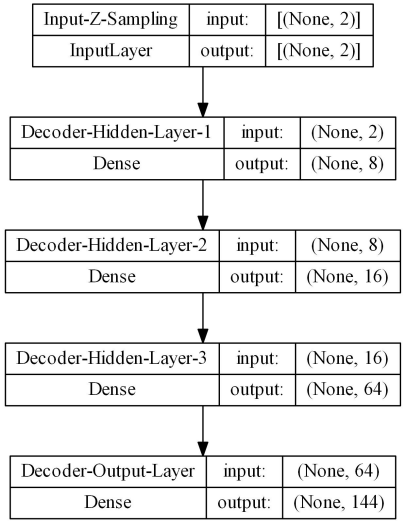
Variational Auto-Encoder (VAE)

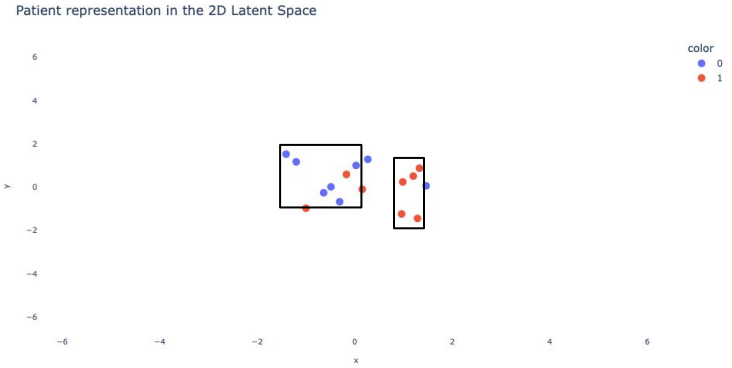
Trouble alert: Small dataset of 50 patients - the majority being High Grade Serous Carcinoma leaves an unbalanced data set and an issue for classification.

Solution: Generating 12x12 pixel pictures of “new” patients from the 2D latent space

The architecture of the VAE



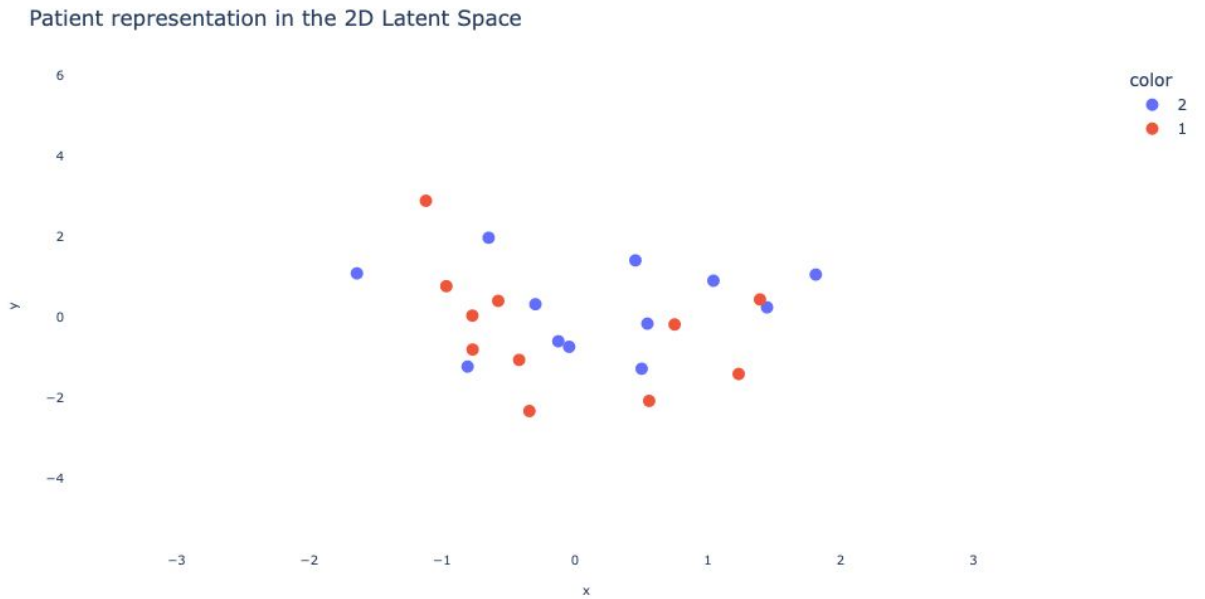




Latent space, multiclass

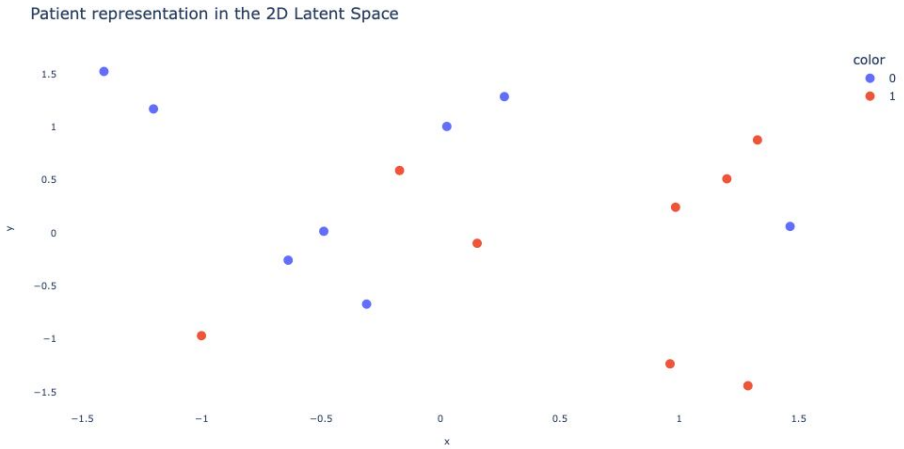
Two classes, High grade serous = 1, Clear cell = 2

Because they cannot be separated in the latent space we decided to not do multi-classification

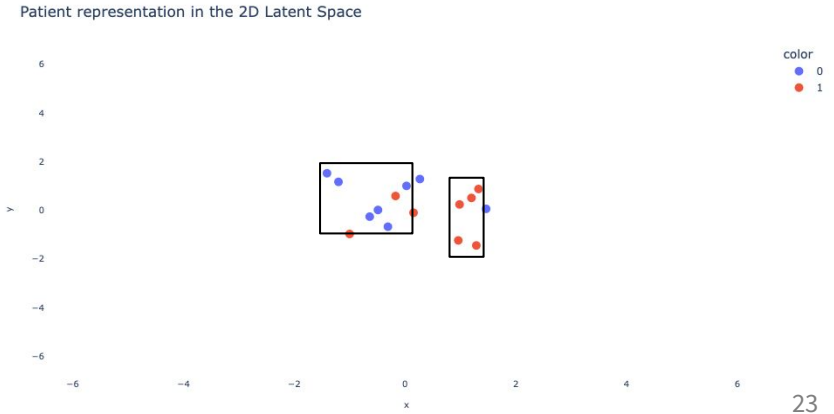


2D Latent Space of binary classification

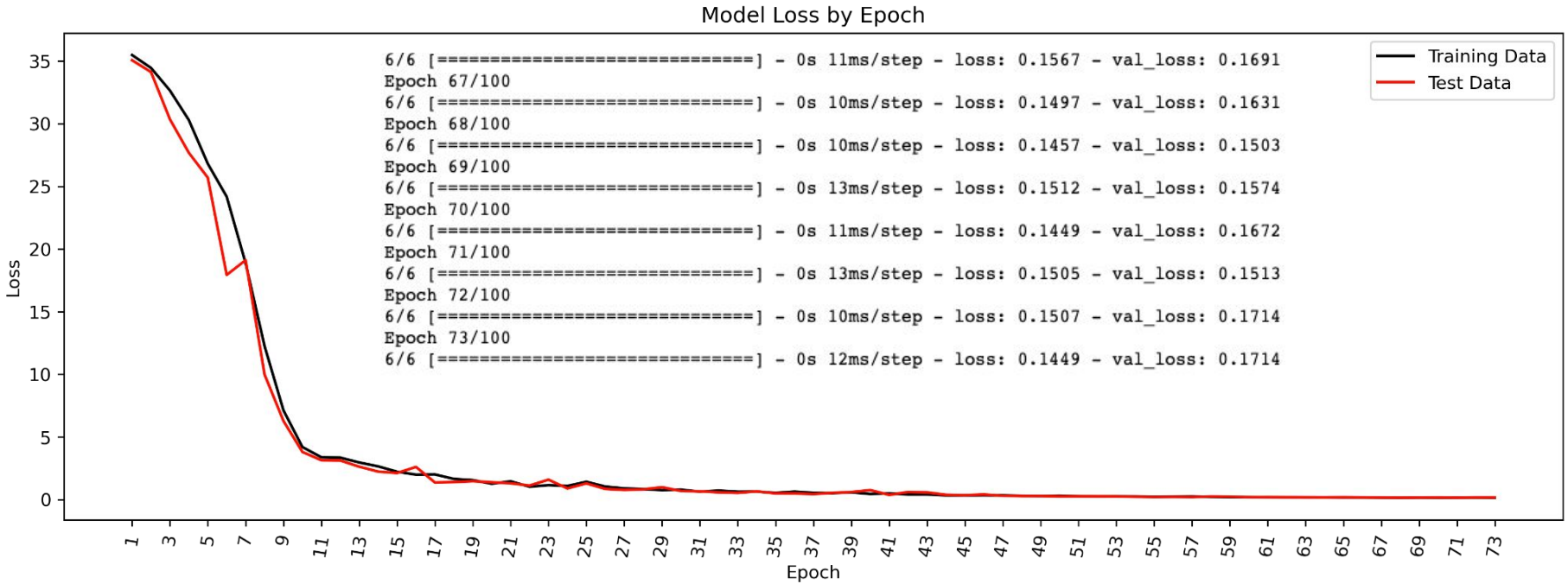
Not best separation of classes, but some kind of separation



It was chosen to simulate new patients within these boxes in the latent space



VAE model training with early stopping.

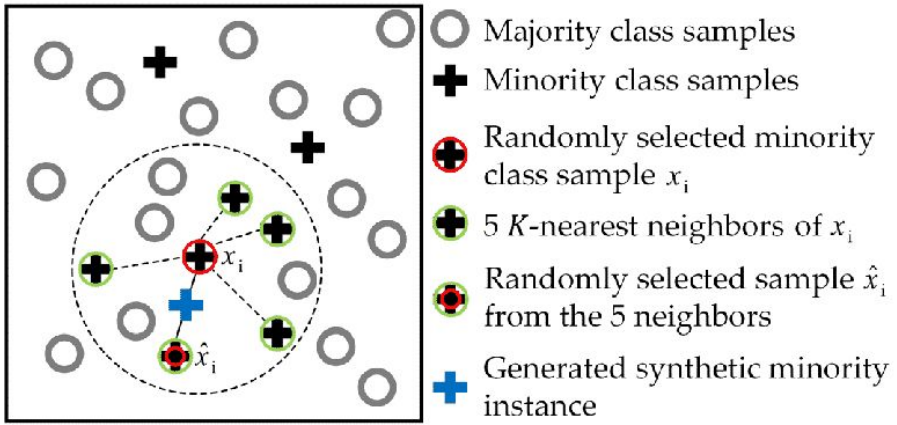


SMOTE oversampling technique

Not satisfied with the first simulation of patients, therefore try SMOTE

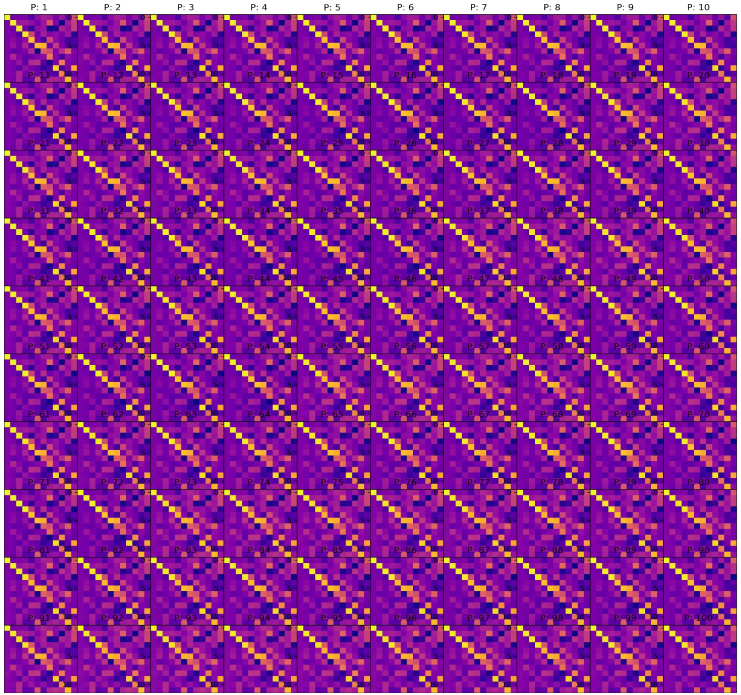
Simulate as many patients as in the biggest group (38) for the other classes.

Simulate more patients in the other classes, will give autoencoder the possibility to train better, hopefully make simulated auto-encoded patients better

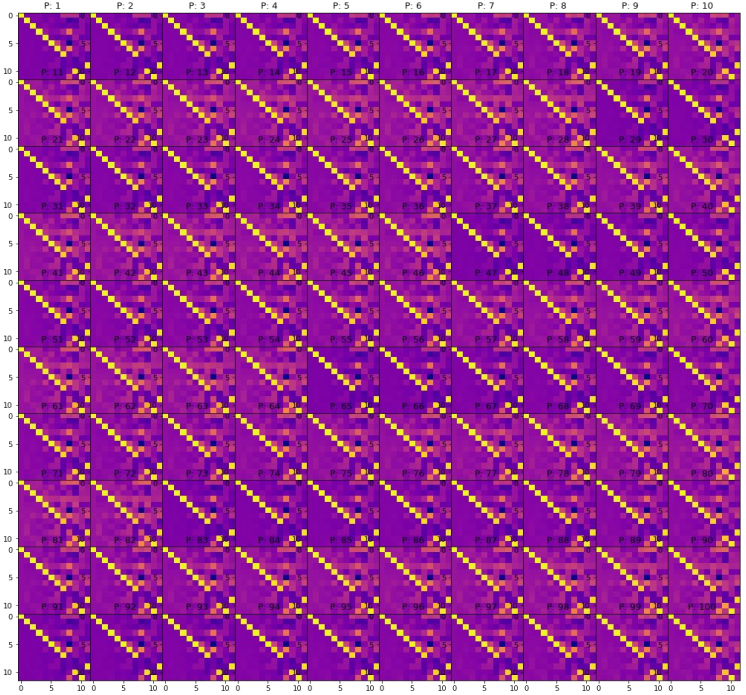


Difference with and without SMOTE

Without SMOTE:
100 simulated patients in the High Grade Serous class

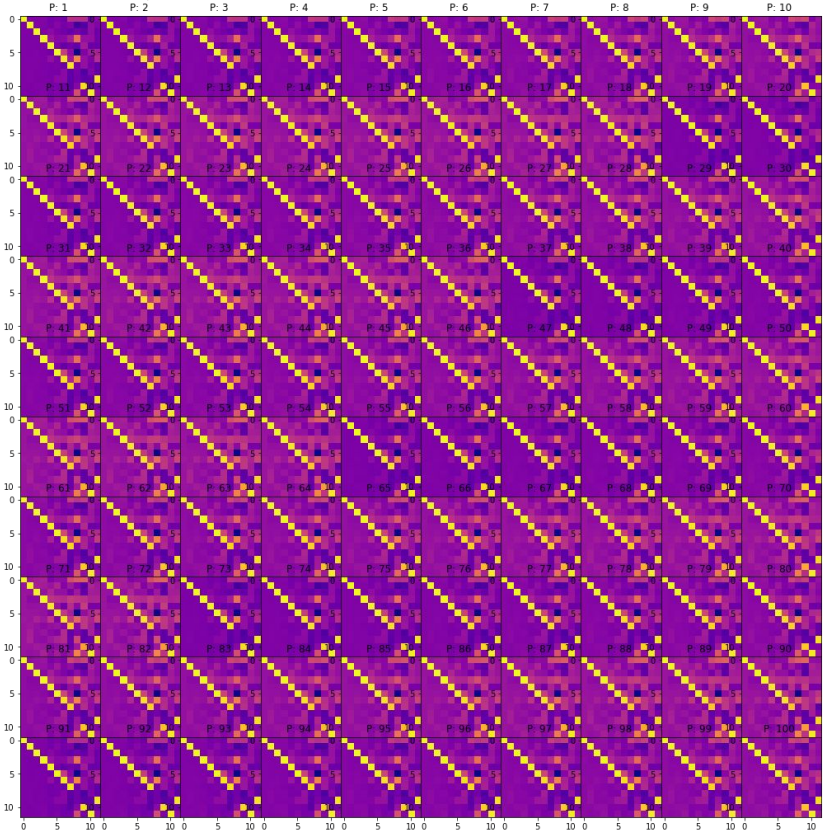


With SMOTE:
100 simulated patients in the High Grade Serous class

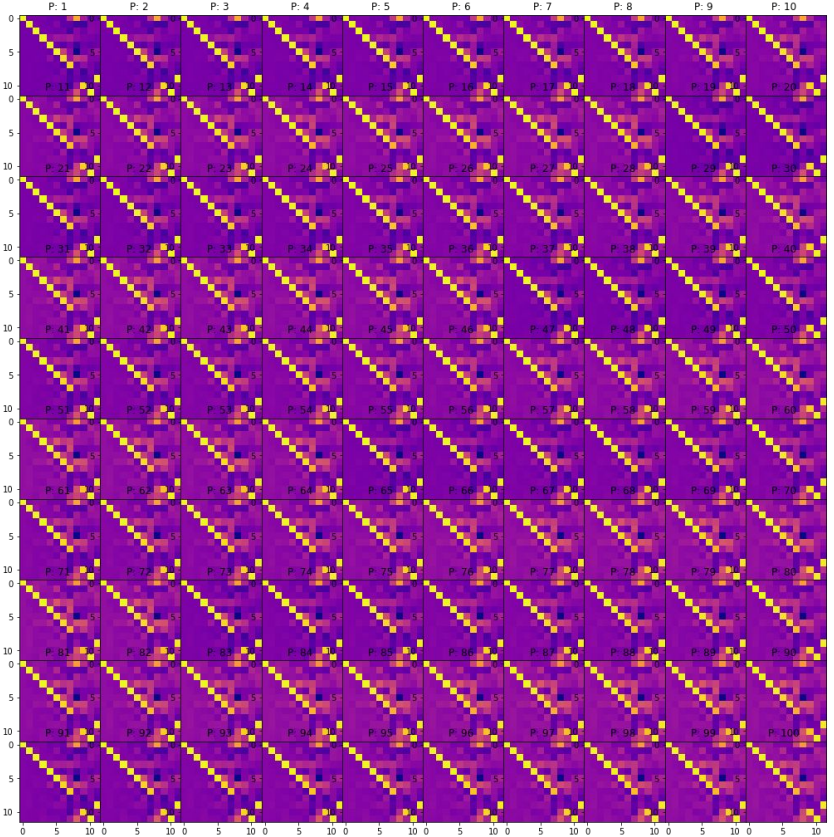


New simulated patients, binary, with SMOTE

Class 1, HGSC



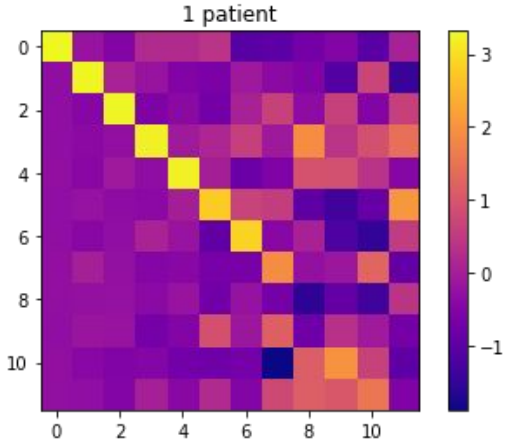
Class 0, Other



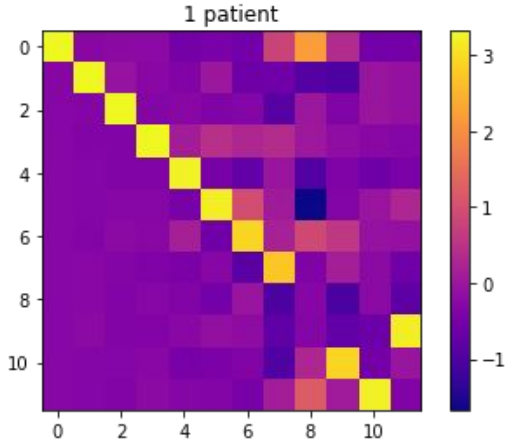
Comparison of performance of VAE with and without SMOTE

Examples of a single patient. - SMOTEing seems to improve simulated patients as it removes some of the random noise around the diagonal line.
When looking at patients a lot of variation lies around the diagonal, so this might be important to replicate.

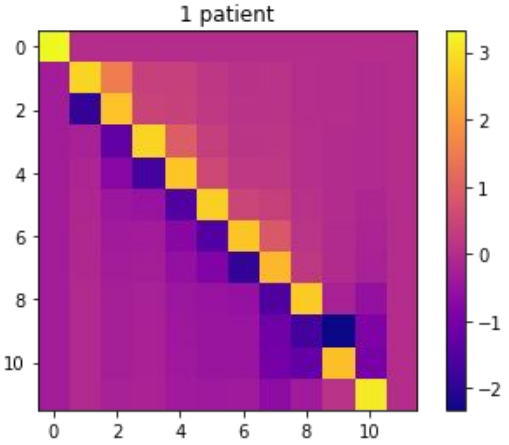
Simulated patient without SMOTE



Simulated patient with SMOTE



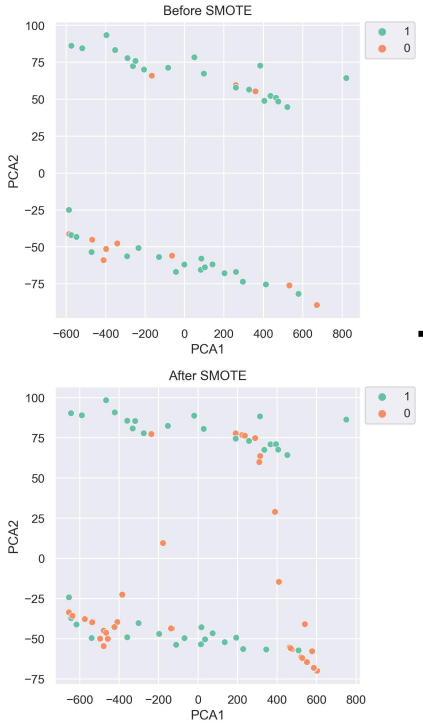
Real patient (target)

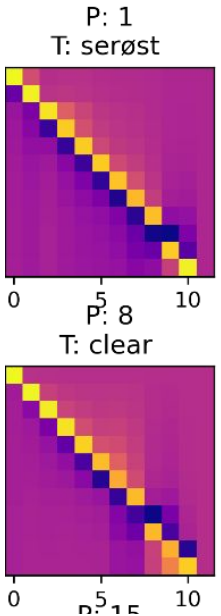


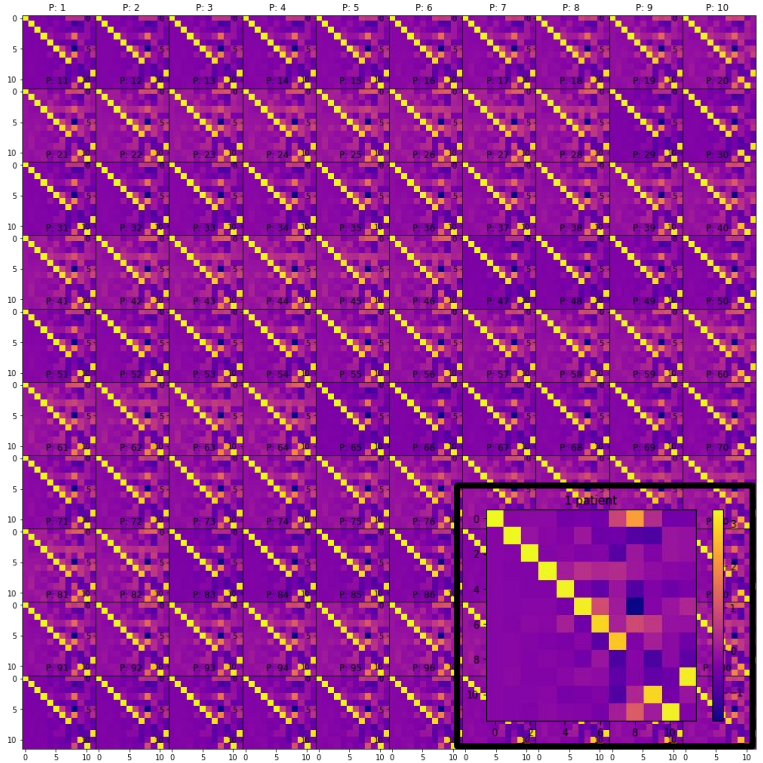
Generate “new” patients/data

Combining Synthetic Minority Oversampling TEchnique and Variational AutoEncoders

- Machine learning models learns poorly when one class dominates the other.
- Little data compromise model performance.







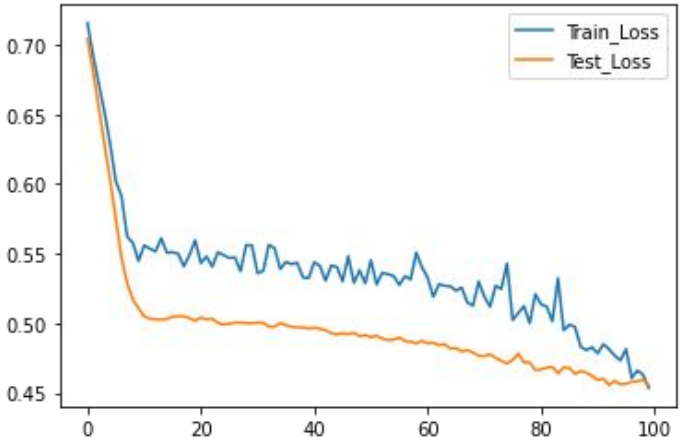
Appendix: CNN

Training CNN without SMOTE

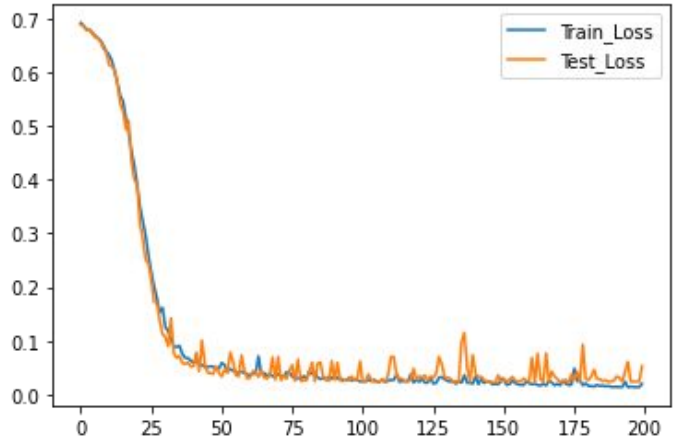
Accuracy scores (test set)	Training and test data: Real patients (49)	Training data: Both simulated and real patients Binary classification (697)	
	Binary (38+11)	Both sim. and real patients in test-set	Only real patients in test-set
CNN	80%	98%	80%

Loss plots have many small spikes, this might hint that the batch size could be optimized - as too small batch sizes might not span all the different classes

Without sim



With sim

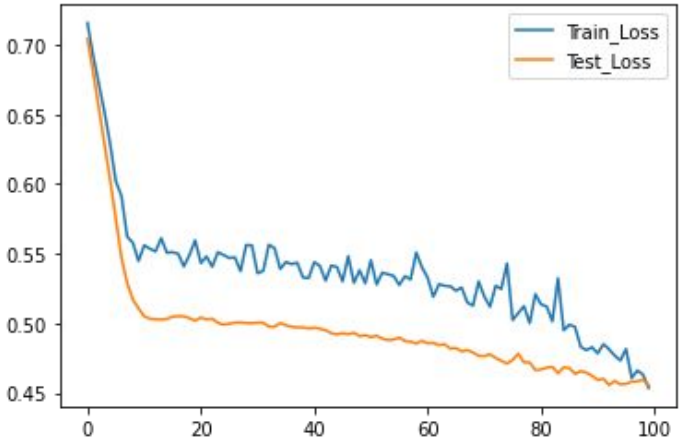


Training CNN with SMOTE

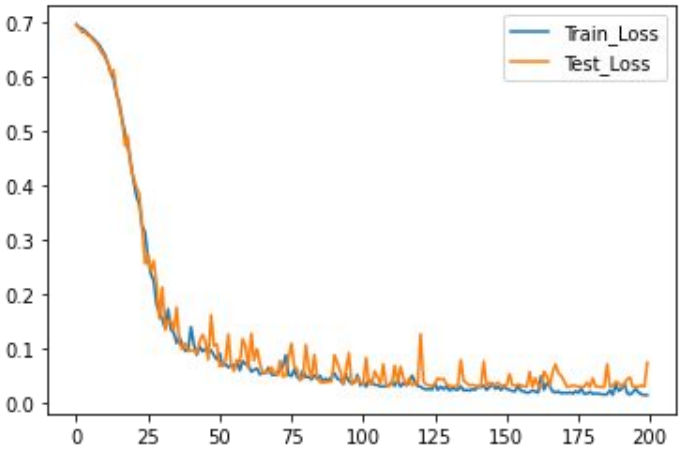
Accuracy scores (test set)	Training and test data: Real patients (49)	Training data: Both simulated and real patients Binary classification (697)	
	Binary (38+11)	Both sim. and real patients in test-set	Only real patients in test-set
CNN	80%	97%	66%

Seems like the CNN performs worse on real patients after addition of smote, however since the test-set was randomized this might be due to chance.

Without sim



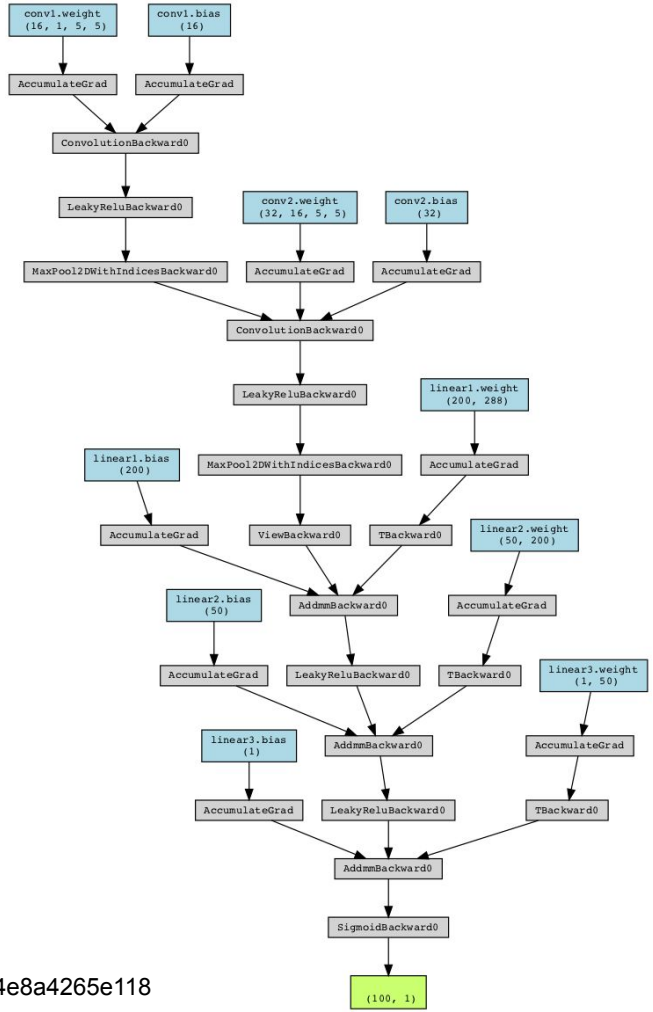
With sim



Full CNN architecture

CNN was programmed in Pytorch based on example code for classification of MNIST. - visualized with Torchviz

- First convolution: 2d Convolution 16 kernels of size 5 -> LeakyReLU
- First maxpool: 2d maxpool kernel size 2
- Second convolution: 2d Convolution 32 kernels of size 5 -> LeakyReLU.
- Second maxpool: 2d maxpool size 2
- reshaped to batchsize x 288
- passed through NN with: 200 -> 50 nodes and either 4 or 1 output.
- Last activation function is Sigmoid



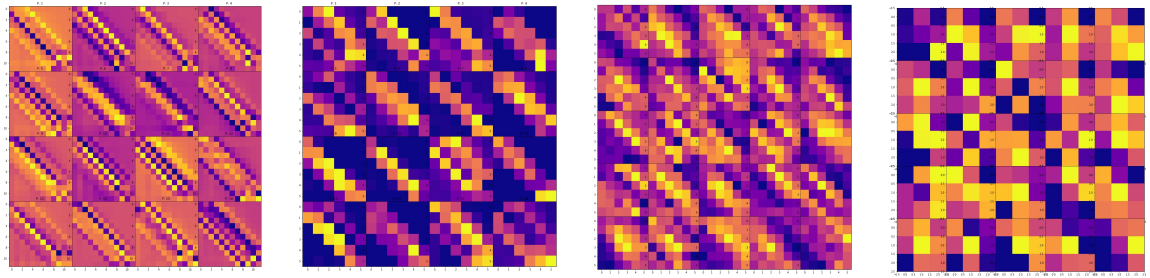
Architecture inspired by:
https://medium.com/@nutanbhogendrasharma/pytorch-convolutional-neural-network-with-mnist-dataset-4e8a4265e118

CNN “kernels” - Plots of a single PCA passed through each layer

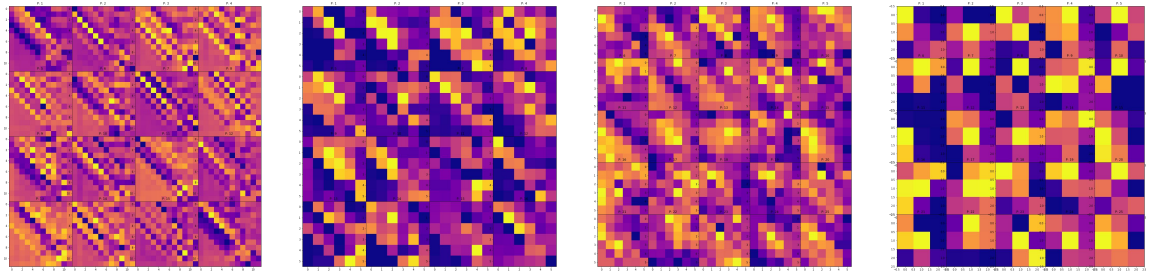
Most seem to accentuate shape of diagonal - which from looking at PCAs as images might be informative, however the trend is difficult to discern. A lot of the structure is visible in the second convolution as well

From left to right: First convolution, after first maxpool, after second convolution, after second maxpool.

Trained on real patients



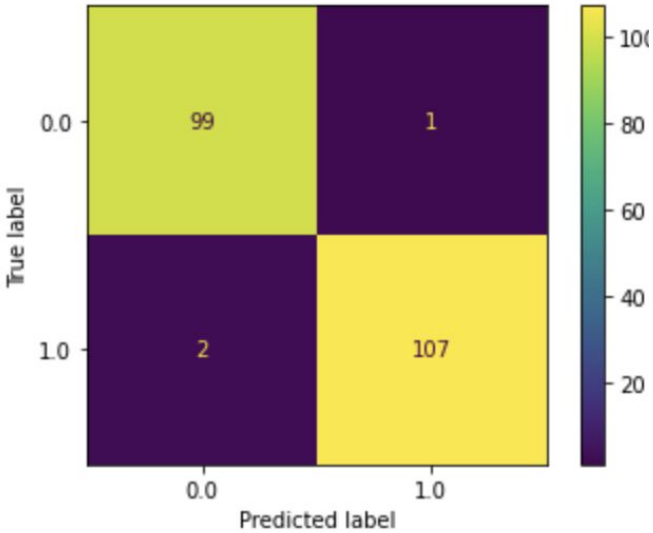
Trained on real + sim



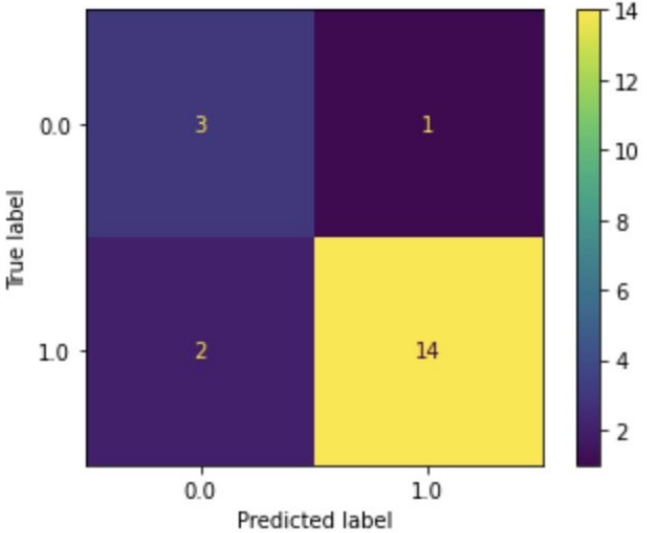
Appendix: Light GBM

Confusion matrix from LGBM models

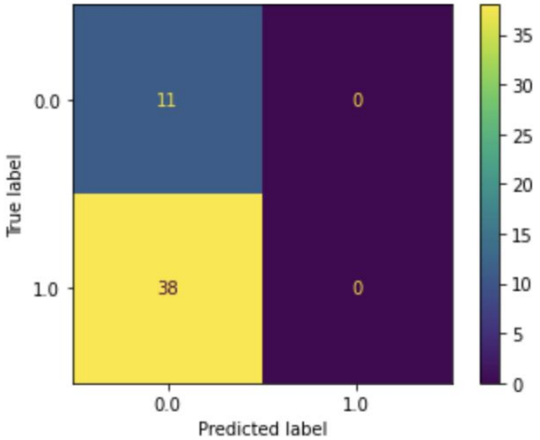
Train: Both sim. and real patients
Test: Both sim and real patients



Train: Both sim. and real patients
Test set: Only real patients



Train: Sim. patients
Test: Real patients



All are predicted as class 0, even though most of them really are class 1. They look more like the simulated class 0.

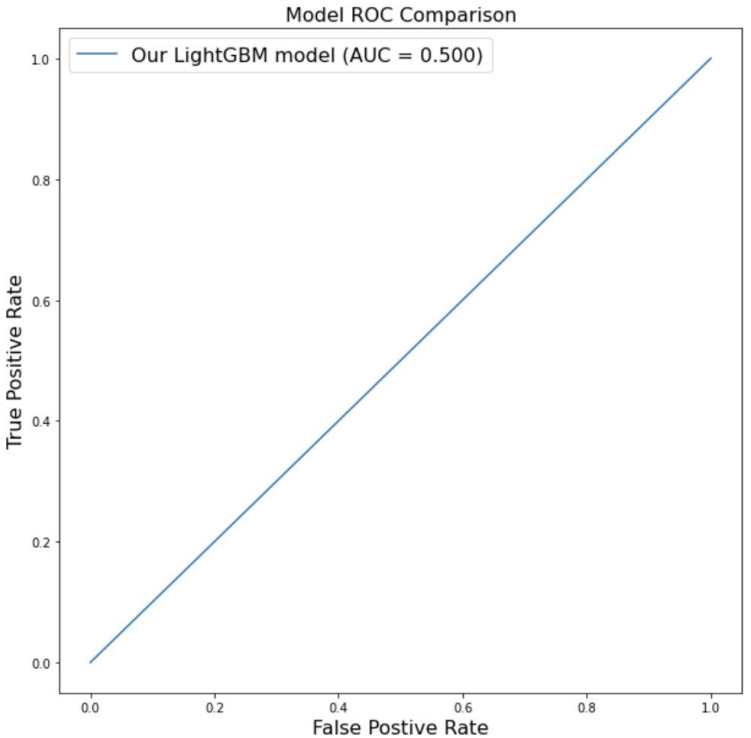
Boosted Decision Tree on real patients (49)

- LightGBM
- Nothing done about the very unbalanced distribution of patients in cancer subtype.
- Hyperparameter optimization by random search, 5 fold CV
- Resulted in max_depth: 23, samples_leaf: 72 and best accuracy score: 0.775
- Same for both multi-classification and binary classification
- Multi-classification ended up being binary, test set consisted of 9 samples, only 2 different classes represented.

Boosted Decision Tree on real patients (49)

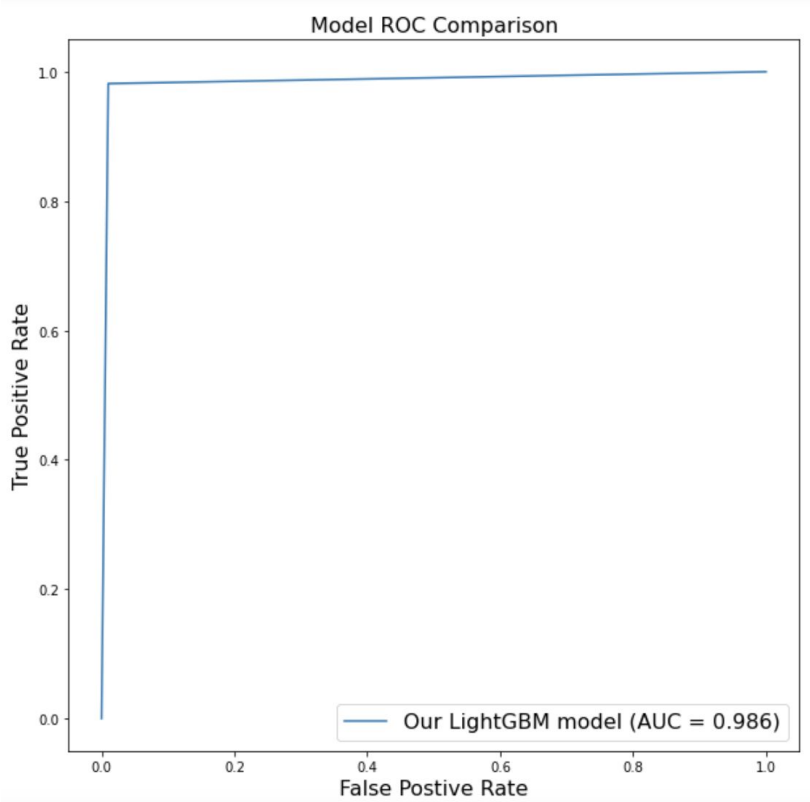
- ROC curve shows that this classifier is extremely bad, it is guessing. This can be explained by the overrepresentation of high-grade serous patients
- Looking into the predictions, for both binary and multi-class the whole test set was predicted as high-grade serous

ROC curve for binary classification



Boosted Decision Tree on sim. and real patients (697)

- Binary classification with LightGBM
- Train and test: Both real and simulated patients
- Random search for hyperparameter-optimization
- Resulted in max_depth: 23, samples_leaf: 1 and best accuracy score: 0,979
- Accuracy on test set: 0,9856

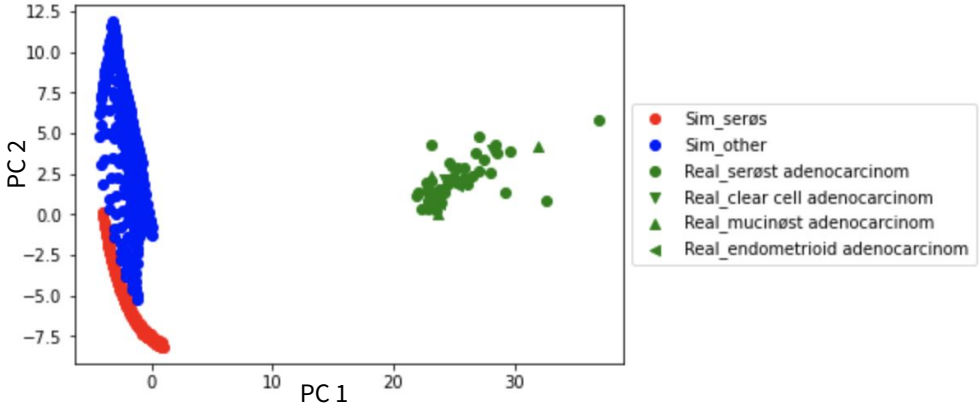


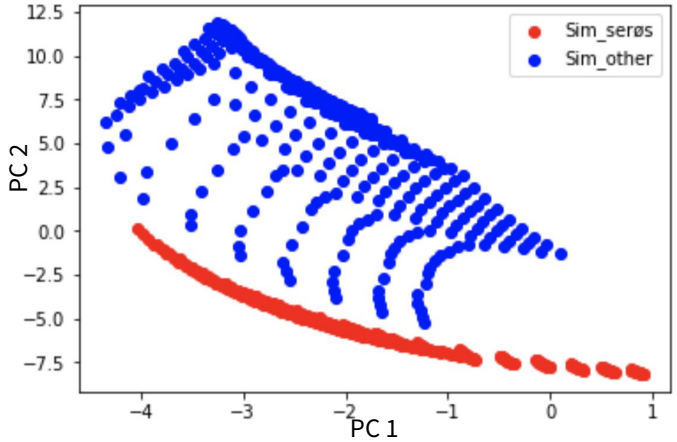
Why did the classification perform like this, LGBM

- To investigate: PCA on the input data for the LGBM
- Clear to see the real patients are very different from the simulated patients
- Have moved slightly away from the real patients in the simulation
- PCA on only simulated patients, no overlapping, therefore easy to differentiate and thereby classify.

To investigate the difference further:

- LGBM trained only on sim. patients, test set of only real patients, accuracy score = 22%
- Predicting all real patients as class “other”

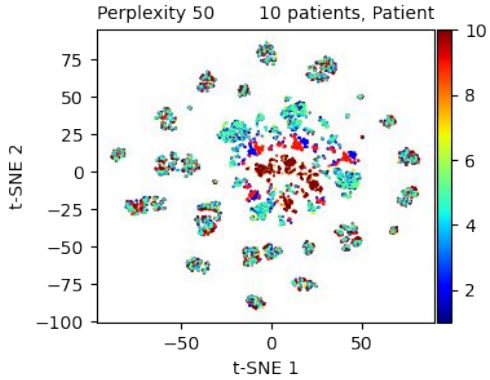


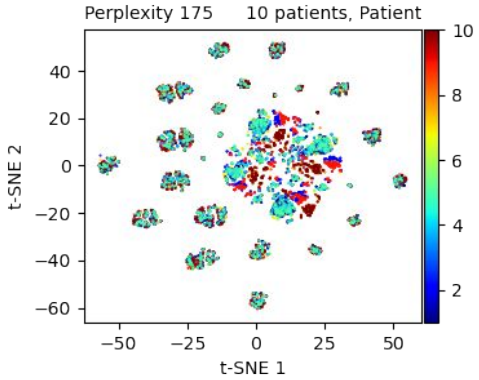


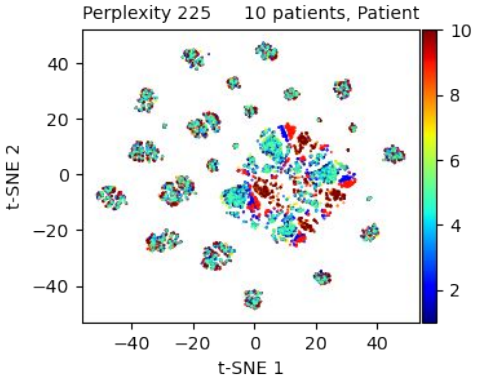
Appendix: Dimensionality reduction and clustering

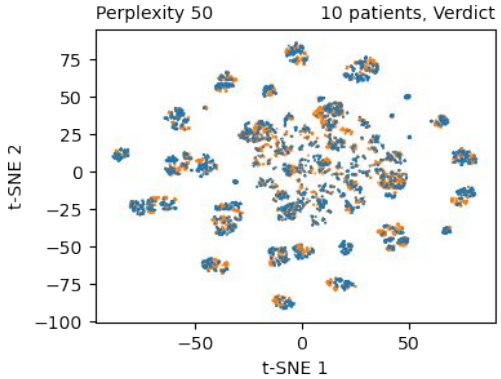
t-SNE hyper parameter optimization

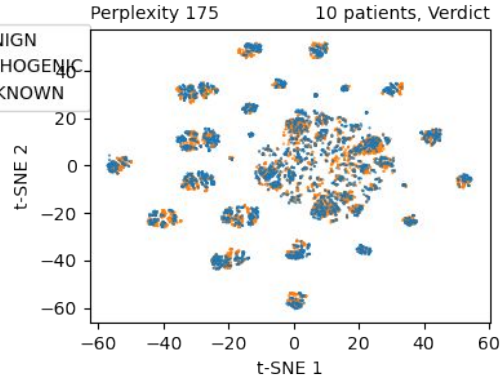
Optimizing perplexity in the range from 50 to 225 - done on a subset of the data.
A difference in the clustering is seen especially in the big cluster in the middle of the plots.

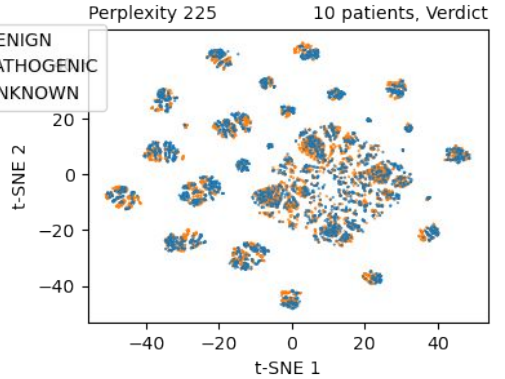


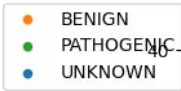


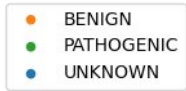








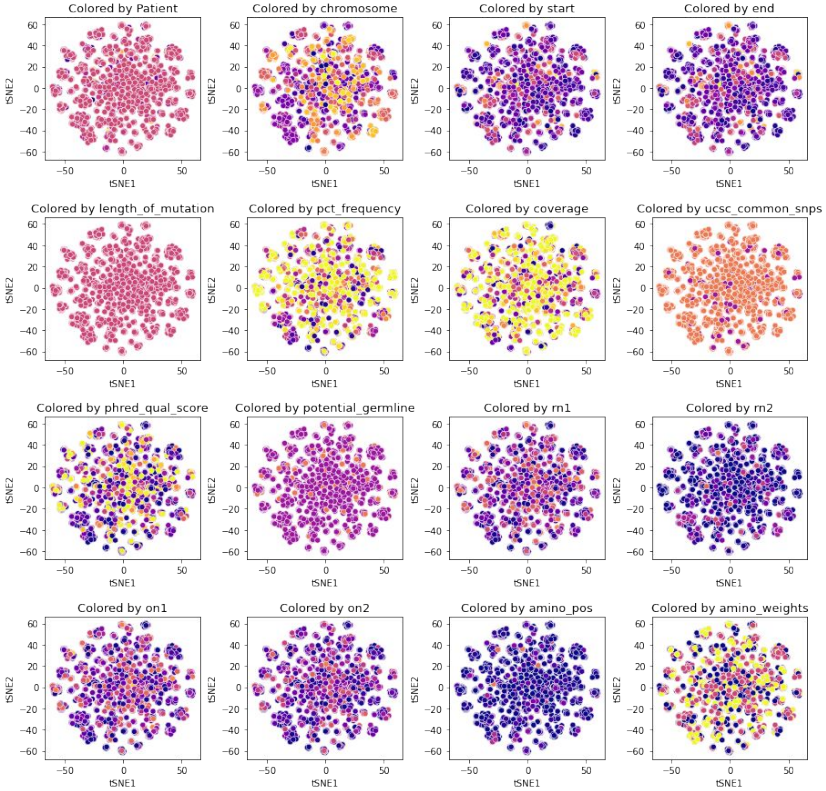




t-SNE colored by different features

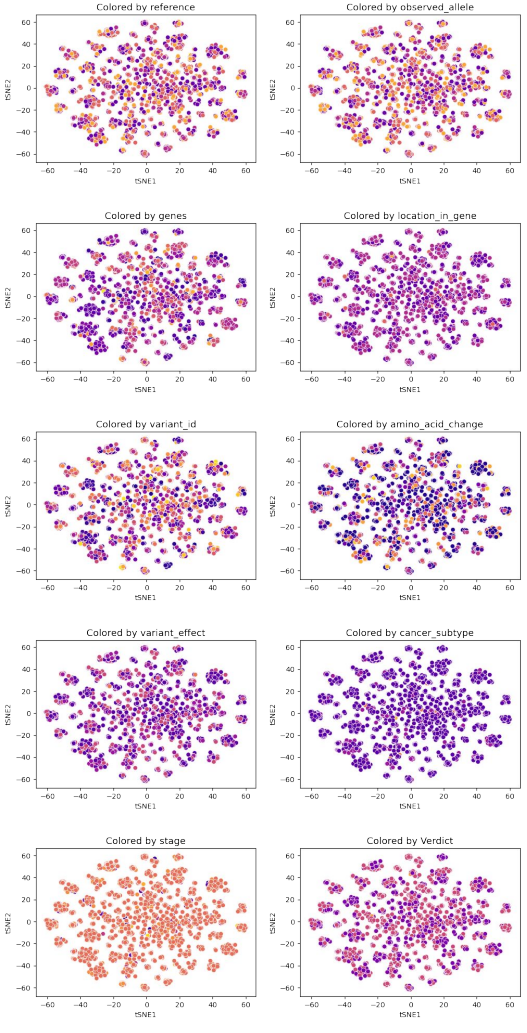
No apparent clustering is visible based on the different features

Discrete feature values



t-SNE mappings colored by some of the different meta data features

Some of them are more informative than others. Seems that the discrete values are more informative in this projection



t-SNE on different number of patients (10 and 50)

The t-SNE projection had a hard time working on the complete dataset compared to a small fraction of the dataset - So we optimized on a reduced dataset - however we may need to rerun optimization to get as clear and informative clusters.