Using machine learning algorithms to analyze RNA-Seq gene expression data for breast cancer biomarkers

Data Representation

Condition	No. of samples
Healthy	112
Luminal A	560
Luminal B	209
HER2	82
Basal	188
Normal-Like	39

Dataset:
Gene Expression Matrix

Gene ID

Sample ID

Gene expression value

```
In [6]: LumA
```

Out[6]:

	ENSG00000000003.15	ENSG00000000005.6	ENSG00000000419.13	ENSG00000000457.14	ENSG00000000460.17	ENSG00000000938.13	ENSG0000000097
TCGA-AC-A8OP-01A-11R-A36F-07	4.102683	-0.264710	5.295867	4.722433	2.622071	2.468591	5.25
TCGA-D8-A1XU-01A-11R-A14M-07	6.665188	-2.768219	5.714447	4.453470	2.636545	2.603979	5.07
TCGA-BH-A18L-11A-42R-A12D-07	6.207314	5.266865	4.878837	4.308785	1.970811	3.559367	7.76
TCGA-BH-A0B1-01A-12R-A056-07	6.008083	-2.925248	5.258046	4.925308	3.735960	2.184225	6.30
TCGA-3C-AAAU-01A-11R-A41B-07	3.600208	-3.269989	5.223848	4.188995	2.867243	2.148495	4.40
...	...	...	...	...	...	...	
TCGA-E2-A15M-11A-22R-A12D-07	6.467658	3.216707	5.185411	3.972429	2.266910	3.735292	7.93
TCGA-E9-A1RC-01A-11R-A157-07	6.731206	-1.674202	5.397395	5.581763	3.920658	1.831691	3.29
TCGA-GM-A2DN-01A-11R-A180-07	5.707511	-3.134850	4.786142	5.478670	4.056665	4.182747	5.63
TCGA-EW-A1IW-01A-11R-A13Q-07	4.947886	-0.400925	4.936676	4.874485	3.944461	2.704657	5.80
TCGA-BH-A42V-01A-11R-A24H-07	5.257250	-0.238640	4.916415	4.463456	2.444224	2.644328	6.37

672 rows × 24888 columns

Principal Component Analysis

- Reduces dimensionality of data set

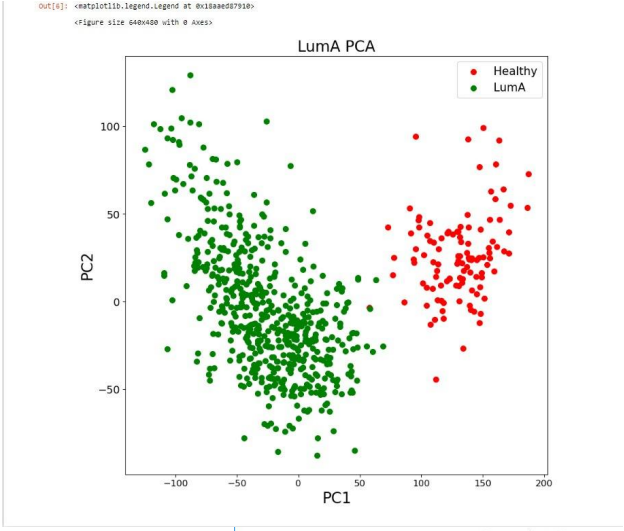
```
#PCA
pca = PCA(n_components=50)
principalComponents_LumA = pca.fit_transform(LumA)
principal_LumA_Df = pd.DataFrame(data = principalComponents_LumA
             , columns = ["PC"+str(i) for i in range(1, 51)])
```

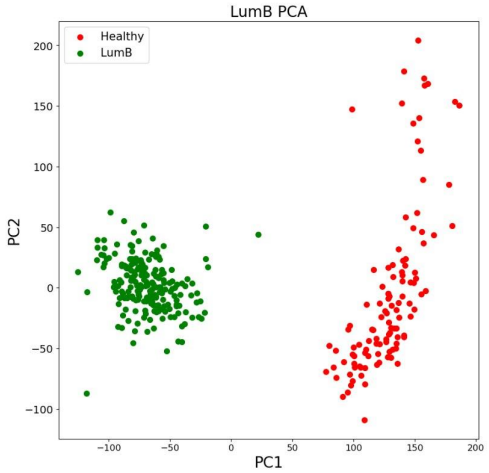
- Principal components are constructed as linear combinations of original variables
- They represent the directions of the data that explain a maximal amount of variance

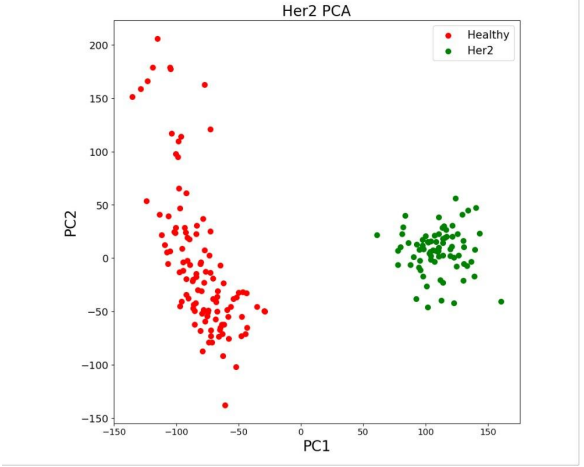
- First principal component accounts for the largest possible variance
- Second principal component accounts for the second largest variance

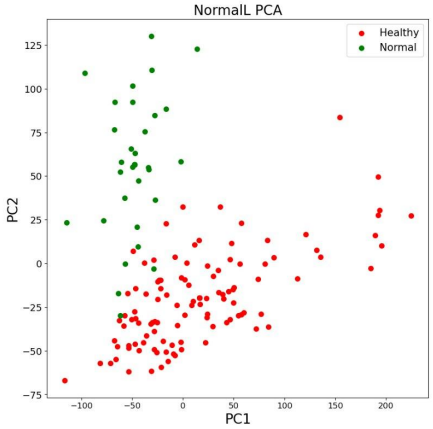
	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	...	PC41	PC42	PC43
0	-34.756513	-3.897099	-3.866967	-37.878473	23.302470	27.181278	-2.119839	19.179626	-3.835512	-2.736174	...	-11.510567	3.640228	-2.530347
1	-39.140790	-15.578549	-19.478732	3.703368	1.989464	-4.780741	-5.966284	27.736212	-17.946515	30.352194	...	25.668447	-11.047957	-0.636587
2	141.021287	24.722809	15.153526	9.227828	-1.213893	1.005873	-0.140307	7.004252	-7.869406	29.316503	...	2.941213	0.133985	-1.324682
3	-28.016296	-7.519097	5.767009	-4.948354	-13.112142	17.449384	-15.706519	19.766532	-5.308975	-18.779413	...	-15.570724	4.630780	14.670408
4	-74.999728	56.568584	-12.983968	-41.982581	52.447397	15.198100	-37.483455	8.330549	18.272060	42.816601	...	4.076246	-2.366924	-2.177588

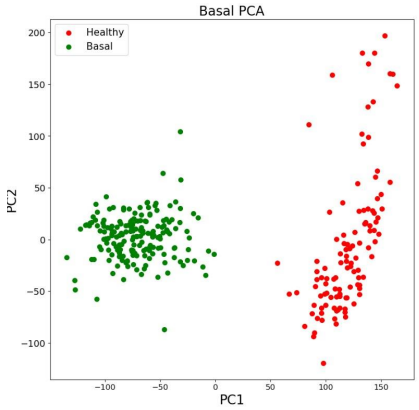
Principal Component Analysis (Results)







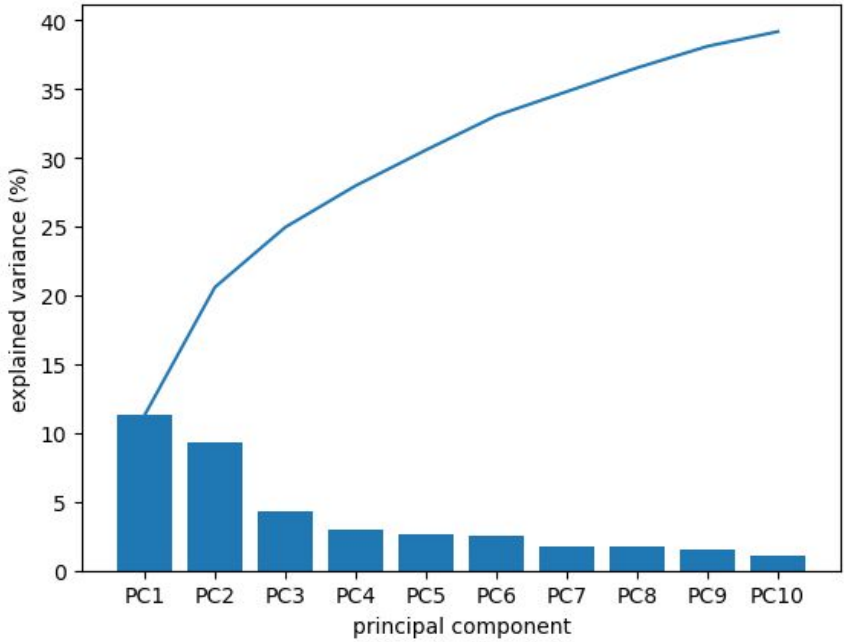




Conclusion: Samples can be differentiated by the gene expression values

```
total=0
for var in pca.explained_variance_ratio_[:10]:
    total+= var
print('First 10 PC accounts for '+str('{0:.3g}'.format(100*total))+'% of variance')
```

```
First 10 PC accounts for 39.2% of variance
```



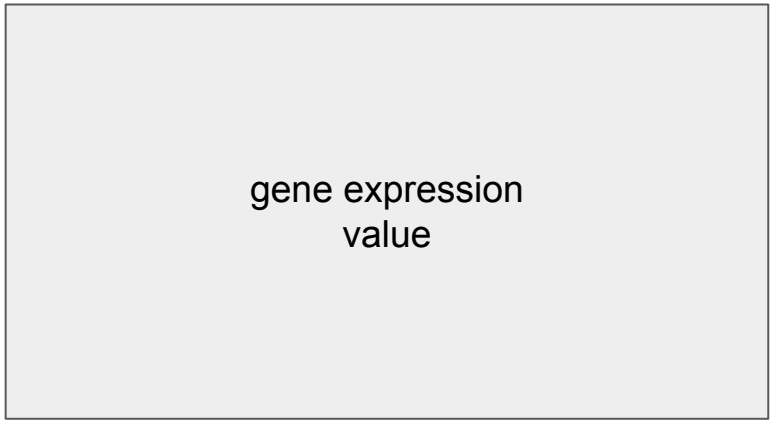
Only 39% of variance could be explained by the first 10 PCs.

Evaluation of PCA

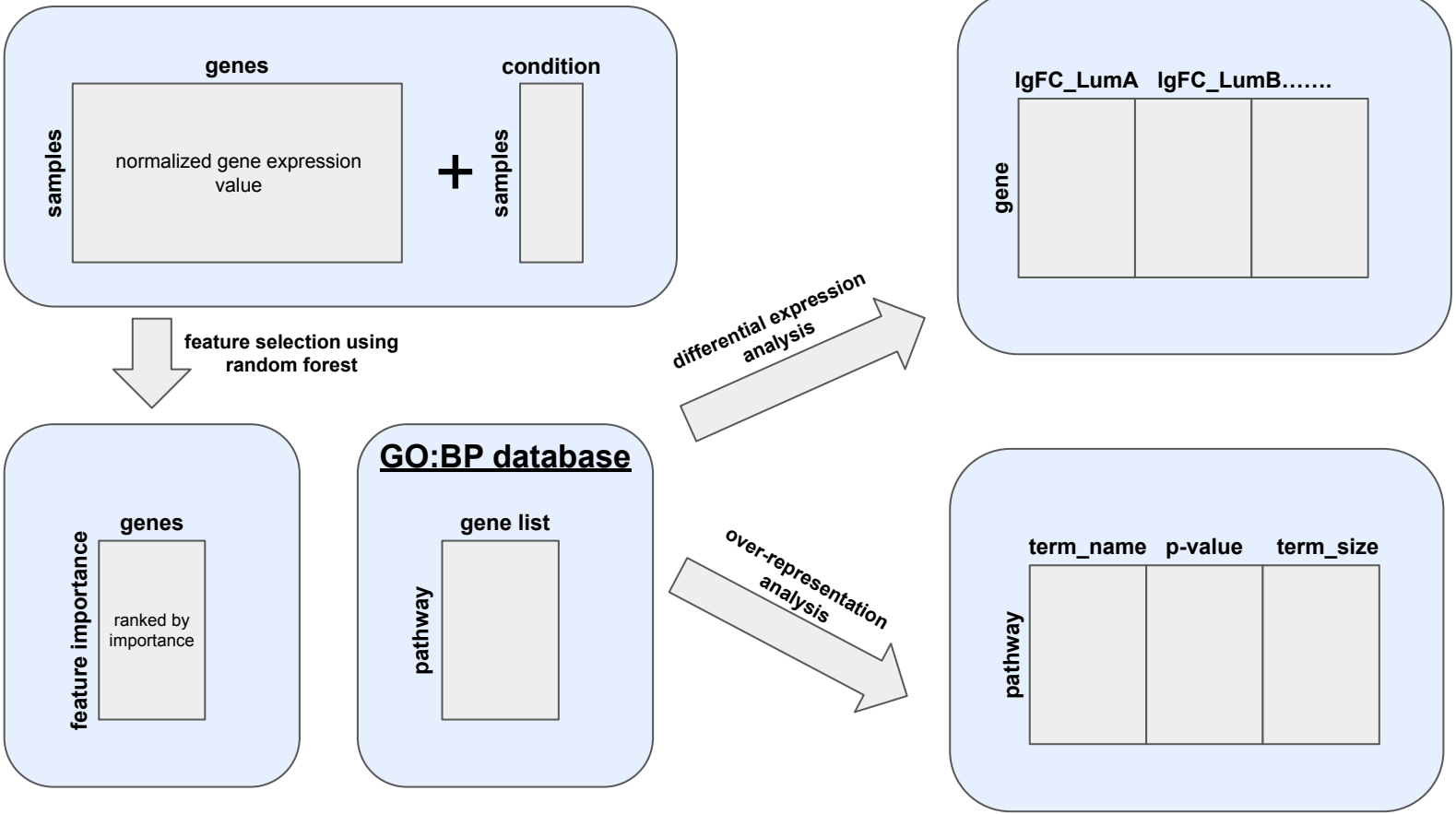
- Assumes linear relationship between features
- Principal components are orthogonal to each other but there might be better basis vectors to represent original features
- Assumes that largest variance is the best way to classify data points

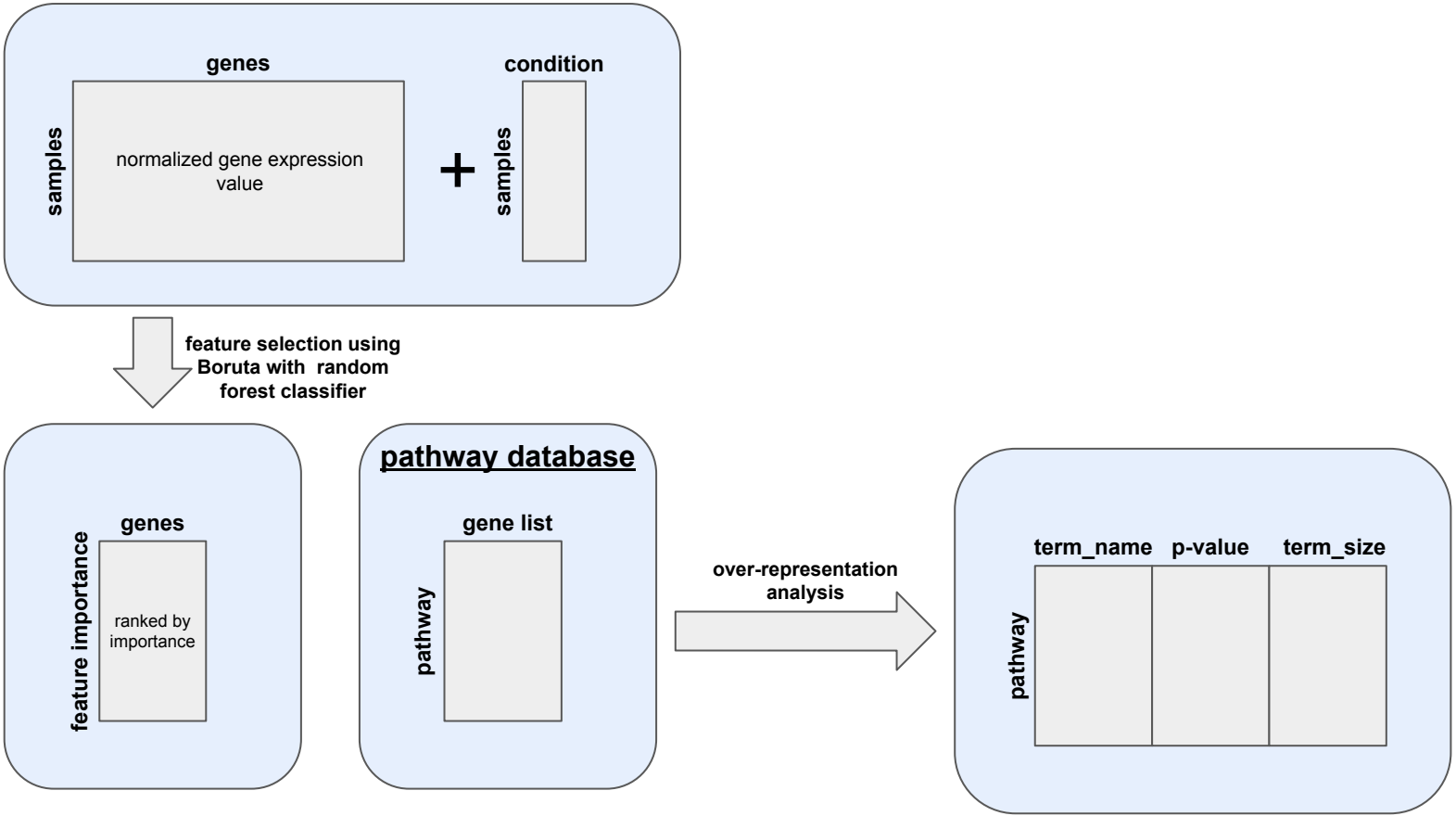
26 991 genes

1190 samples

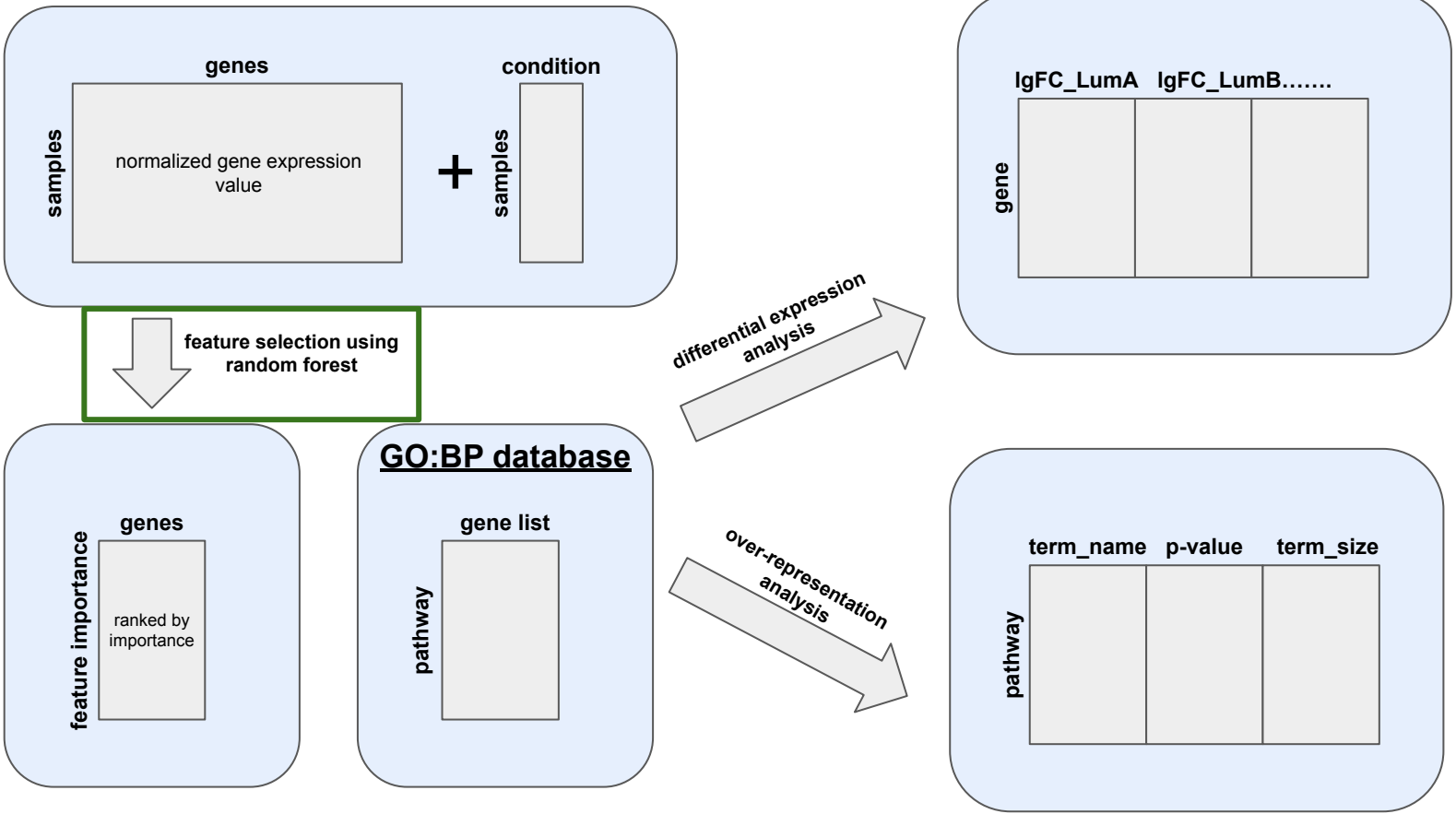


Workflow


genes
condition
samples
normalized gene expression value
+
samples
feature selection using random forest
differential expression analysis
lgFC_LumA
lgFC_LumB…….
gene
genes
feature importance
ranked by importance
GO:BP database
gene list
pathway
over-representation analysis
term_name
p-value
term_size
pathway


genes
condition
samples
normalized gene expression value
+
samples
feature selection using Boruta with random forest classifier
genes
feature importance
ranked by importance
pathway database
gene list
pathway
over-representation analysis
term_name
p-value
term_size
pathway

Workflow

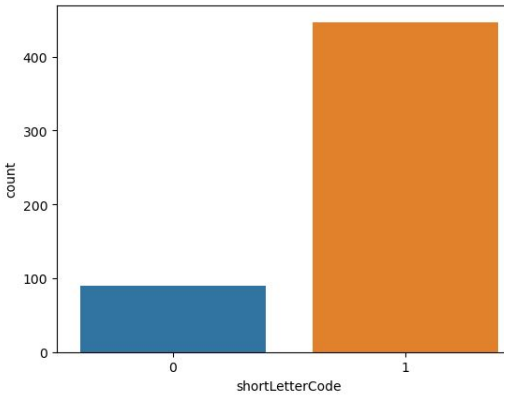

genes
condition
samples
normalized gene expression value
+
samples
feature selection using random forest
differential expression analysis
lgFC_LumA lgFC_LumB…….
gene
genes
feature importance
ranked by importance
GO:BP database
gene list
pathway
over-representation analysis
term_name p-value term_size
pathway

SMOTE to resolve Class Imbalance in Training Data Set

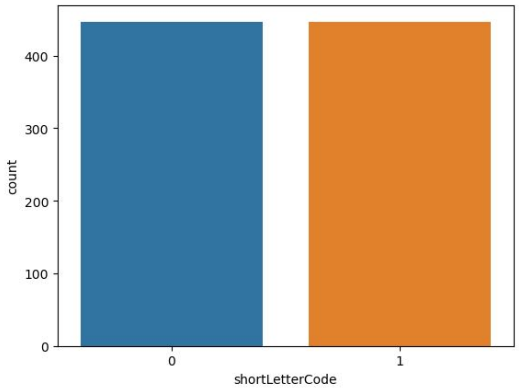
- Resampling method to ensure that number of healthy samples (minority) and tumor samples are the same
- As random forest is a stochastic method, it has some element of randomness
- Few data points would result in higher variance
- After creating more data points by interpolation, variance is reduced

```
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=77)
X, y = smote.fit_resample(X_train, y_train)
```

Before SMOTE



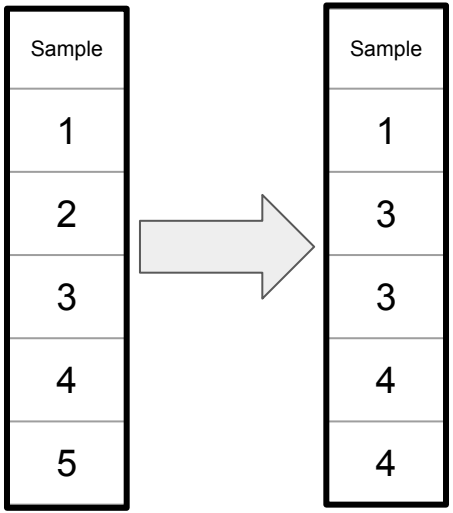
After SMOTE



0 - Primary Solid Tumor
1 - Solid Tissue Normal

Random Forest Classifier

- Samples are selected at random from original data with replacement



- Choose some features at random out of all features, i.e. random subset of genes

√(total no. of features) are chosen

- The best split of these features is used to split the node on the selected samples

- Specified number of decision trees are created (5000 used here)

```
rfc = RandomForestClassifier(random_state=77, n_estimators=1000, max_depth=5)
```

- Outcomes are predicted based on majority voting across all decision trees

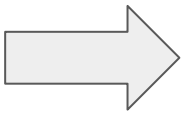
Tree 1	Tree 2	Tree 3	Tree 4	Tree 5	Predicted outcome
1	0	1	1	0	**1**

Feature selection with Boruta for classification

Shadow features

- Shadow features are created by randomly permuting each feature

Feature
1
2
3
4
5



shadow_Feature
3
5
1
2
4

- Random Forest classifier is used on the data set
- Feature importance of each original feature and shadow feature is recorded
- If feature importance > highest feature importance of shadow features, feature gets a hit
- This process is iterated many times e.g. 1000
- Features are classified into confirmed, tentative and rejected features according to binomial distribution

```
rfc = RandomForestClassifier(random_state=77, n_estimators=1000, max_depth=5)
boruta_selector = BorutaPy(estimator=rfc, n_estimators=1000, verbose=2, random_state=77, max_iter=1000, alpha=0.0
boruta_selector.fit(np.array(X_train), np.array(y_train))
```

```
Iteration:      1 / 1000
Confirmed:      0
Tentative:      24888
Rejected:       0
Iteration:      2 / 1000
Confirmed:      0
Tentative:      24888
Rejected:       0
Iteration:      3 / 1000
Confirmed:      0
Tentative:      24888
Rejected:       0
Iteration:      4 / 1000
Confirmed:      0
Tentative:      24888
Rejected:       0
```

```
Iteration:      998 / 1000
Confirmed:      395
Tentative:      173
Rejected:       24320
Iteration:      999 / 1000
Confirmed:      395
Tentative:      173
Rejected:       24320

BorutaPy finished running.

Iteration:      1000 / 1000
Confirmed:      395
Tentative:      113
Rejected:       24320
```

Feature selection with Boruta

```
selected_rf_features = pd.DataFrame({'Feature':list(X_train.columns),
                                     'Ranking':boruta_selector.ranking_})
selected_rf_features.sort_values(by='Ranking')
```

Only genes with ranking 1 are selected

```
selected_boruta_genes = ranked_boruta_genes.query('Ranking == 1')
```

	Feature	Ranking
9720	ENSG00000160753.16	1
9052	ENSG00000154678.18	1
18005	ENSG00000230838.1	1
13022	ENSG00000178662.16	1
1369	ENSG00000078596.11	1
...	...	...
12259	ENSG00000173402.12	17419
12261	ENSG00000173406.16	17419
1498	ENSG00000082397.17	17419
1495	ENSG00000082258.13	17419
14132	ENSG00000185800.12	17419

24888 rows × 2 columns

	Feature	Ranking
9720	ENSG00000160753.16	1
9052	ENSG00000154678.18	1
18005	ENSG00000230838.1	1
13022	ENSG00000178662.16	1
1369	ENSG00000078596.11	1
...	...	...
13739	ENSG00000183682.8	1
6992	ENSG00000137731.14	1
7332	ENSG00000139874.6	1
6511	ENSG00000135046.14	1
6226	ENSG00000132970.14	1

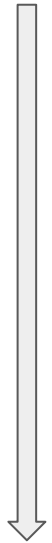
395 rows × 2 columns

Genes are then sorted by importance

```
importance.sort_values(by='importance',ascending=False)
```

	Feature	importance
88	ENSG00000149231.15	0.041038
119	ENSG00000243742.5	0.039784
207	ENSG00000173890.17	0.038569
276	ENSG00000129354.12	0.036450
290	ENSG00000114547.10	0.036420
...	...	...
126	ENSG00000166265.13	0.000012
366	ENSG00000136158.12	0.000010
96	ENSG00000262097.2	0.000000
354	ENSG00000225889.9	0.000000
349	ENSG00000133800.9	0.000000

395 rows × 2 columns



importance

Evaluation of Boruta Feature Selection

- Using only selected features to perform classification on test dataset

  Accuracy Score = no. of samples correctly classified / no. of total samples

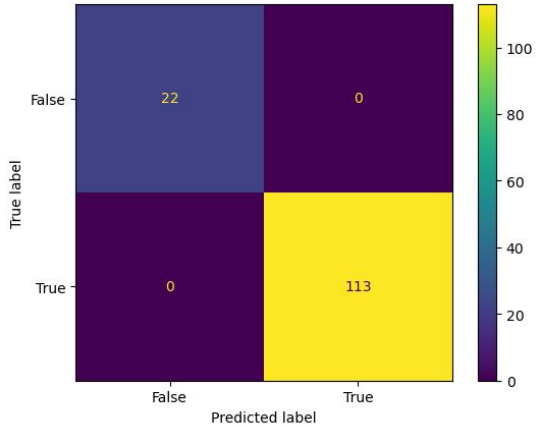
```
In [13]:  1  accuracy_score(y_test, rf_boruta.predict(X_important_test))

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    0.2s finished

Out[13]: 1.0
```

- F1 score

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

- Confusion matrix

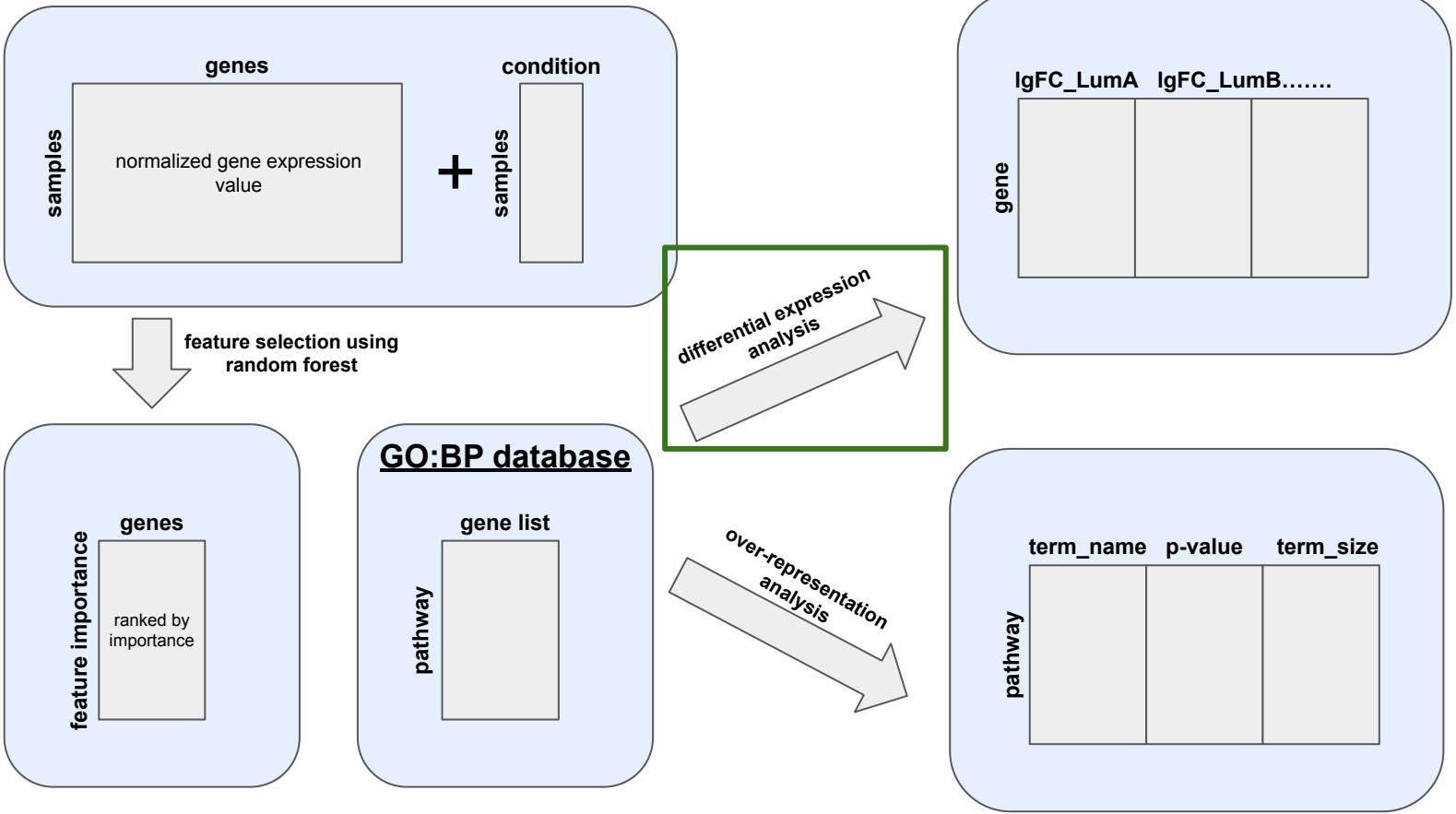
```
1 confusion_matrix = metrics.confusion_matrix(y_test, rf_boruta.predict(X_important_test))
2 cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
3 import matplotlib.pyplot as plt
4 cm_display.plot()
5 plt.show()
```




```
1 #F1 score
2 F1_score = metrics.f1_score(y_test, rf_boruta.predict(X_important_test))
3 print({"F1_score":F1_score})
```

```
{'F1_score': 1.0}
```

Workflow

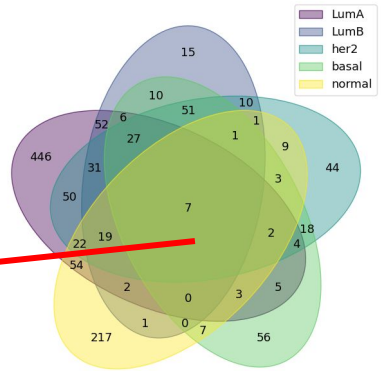

genes
condition
samples
normalized gene expression value
+
samples
feature selection using random forest
differential expression analysis
lgFC_LumA lgFC_LumB…….
gene
feature importance
genes
ranked by importance
GO:BP database
gene list
pathway
over-representation analysis
term_name p-value term_size
pathway

Differential Expression Analysis

- Performed on gene intersections of all 5 molecular subtypes

```
intersection = set(LumA.Feature) &
               set(LumB.Feature) &
               set(her2.Feature) &
               set(basal.Feature) &
               set(normal.Feature)
inter = list(intersection)
```

```
['ENSG00000161649.13',
 'ENSG00000165197.5',
 'ENSG00000186642.16',
 'ENSG00000167434.10',
 'ENSG00000154263.17',
 'ENSG00000168497.5',
 'ENSG00000229246.2']
```



Using gProfiler to obtain gene symbols

```
genes = list(inter_LumA['genes'])
genes_to_convert = [item.split('.',1)[0] for item in genes]
genes_to_convert = '\n'.join(genes_to_convert)
```

```
from gprofiler import GProfiler

gp = GProfiler(return_dataframe=True)
converted = gp.convert(organism='hsapiens',
            query=genes_to_convert,
            target_namespace='ENTREZGENE_ACC')
```

	incoming	name	description
0	ENSG00000161649	CD300LG	CD300 molecule like family member g [Source:HG...
1	ENSG00000165197	VEGFD	vascular endothelial growth factor D [Source:H...
2	ENSG00000186642	PDE2A	phosphodiesterase 2A [Source:HGNC Symbol;Acc:H...
3	ENSG00000167434	CA4	carbonic anhydrase 4 [Source:HGNC Symbol;Acc:H...
4	ENSG00000154263	ABCA10	ATP binding cassette subfamily A member 10 [So...
5	ENSG00000168497	CAVIN2	caveolae associated protein 2 [Source:HGNC Sym...
6	ENSG00000229246	LINC00377	long intergenic non-protein coding RNA 377 [So...

Differential Expression Analysis

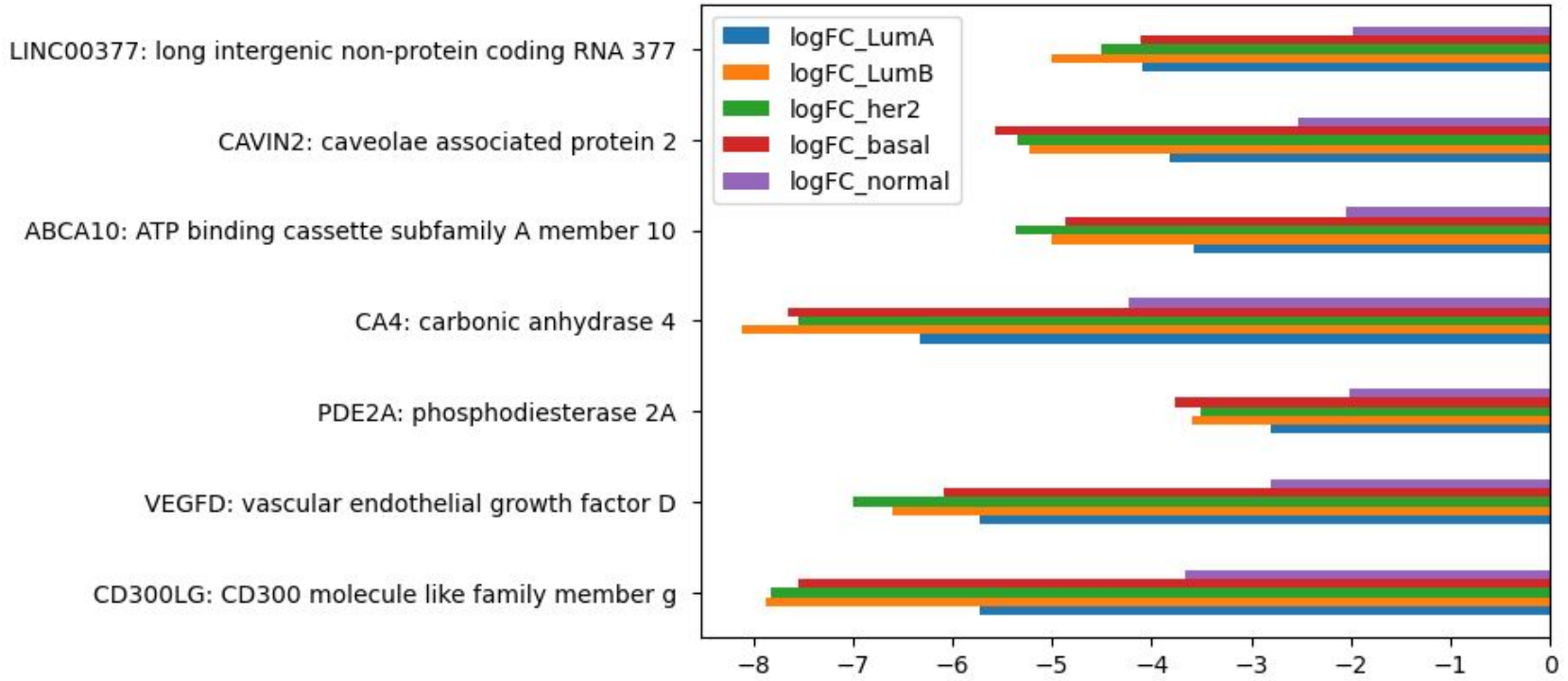
```
names=list(converted.name) #names = gene symbols

descriptions_long=list(converted.description) #long descriptions

descriptions = [item.split('[',1)[0]for item in descriptions_long] # remove [Source...

concat_func = lambda x,y: x + ": " + str(y)
inter_LumA.index = list(map(concat_func,names,descriptions))
inter_LumA.genes = list(names)# list the map function
display(inter_LumA)
```

	genes	logFC_LumA	logFC_LumB	logFC_her2	logFC_basal	logFC_normal
CD300LG: CD300 molecule like family member g	CD300LG	-5.721135	-7.863754	-7.812080	-7.545065	-3.659360
VEGFD: vascular endothelial growth factor D	VEGFD	-5.716690	-6.600323	-6.989058	-6.082338	-2.799917
PDE2A: phosphodiesterase 2A	PDE2A	-2.808472	-3.594250	-3.513679	-3.768122	-2.016734
CA4: carbonic anhydrase 4	CA4	-6.331970	-8.116992	-7.551134	-7.653623	-4.228454
ABCA10: ATP binding cassette subfamily A member 10	ABCA10	-3.580456	-5.009097	-5.360821	-4.862154	-2.054404
CAVIN2: caveolae associated protein 2	CAVIN2	-3.809099	-5.227843	-5.346751	-5.574720	-2.522290
LINC00377: long intergenic non-protein coding RNA 377	LINC00377	-4.093084	-5.003995	-4.499900	-4.109750	-1.974739

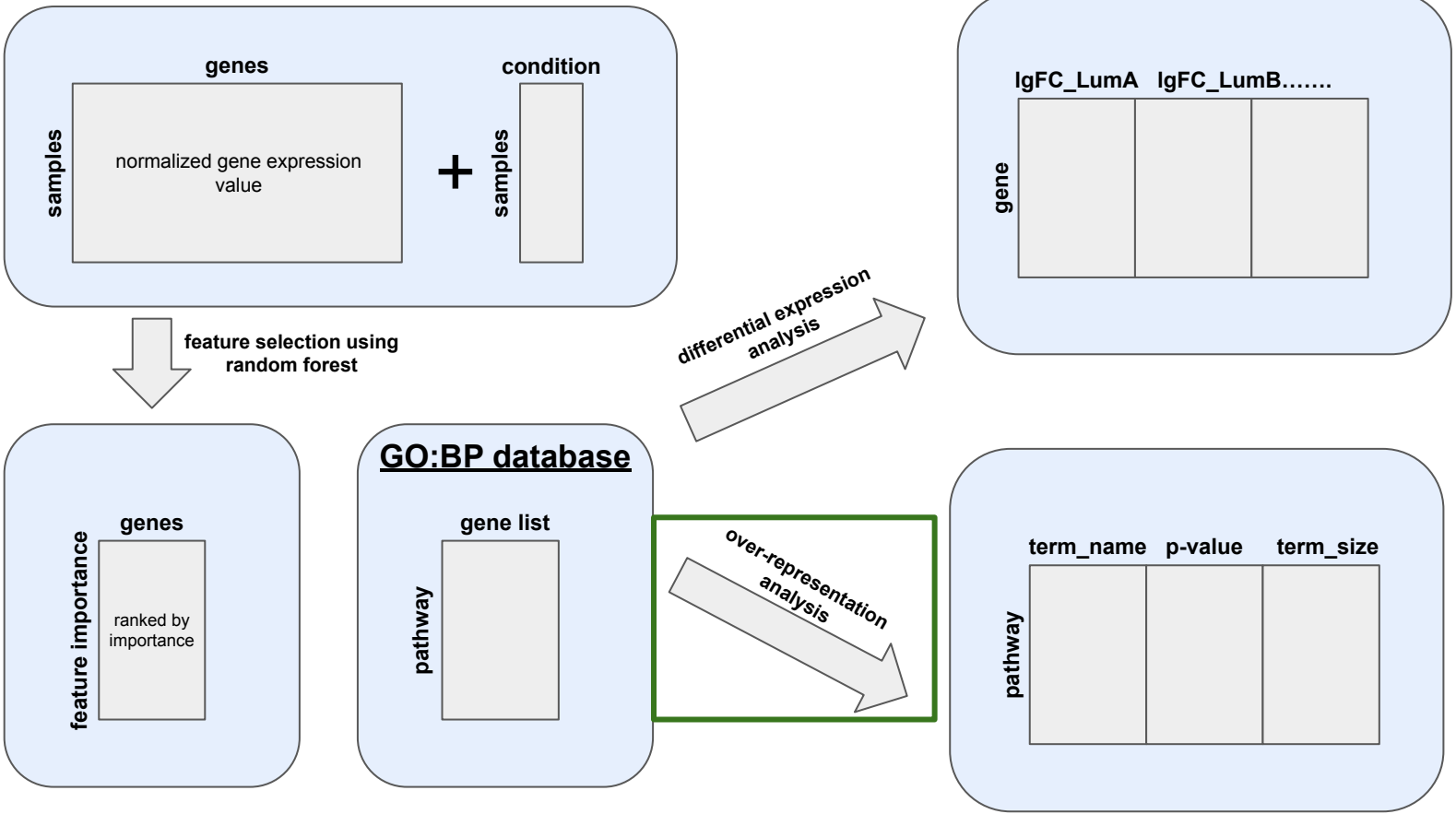
Table of logFC for each gene with respect to all 5 molecular subtypes

Differential Expression Analysis (Results)

```
1 ax = inter_LumA.plot.barh()
2
```



Workflow


genes
condition
samples
normalized gene expression value
+
samples
feature selection using random forest
differential expression analysis
lgFC_LumA lgFC_LumB…….
gene
feature importance
genes
ranked by importance
GO:BP database
gene list
pathway
over-representation analysis
term_name p-value term_size
pathway

Over-Representation Analysis

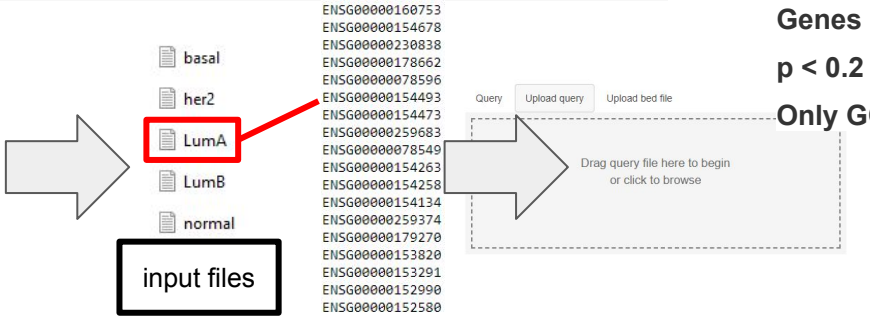
- Determines whether genes from pre-defined sets (e.g. GO terms, KEGG pathways) are present more than would be expected in a set of genes.

gProfiler query

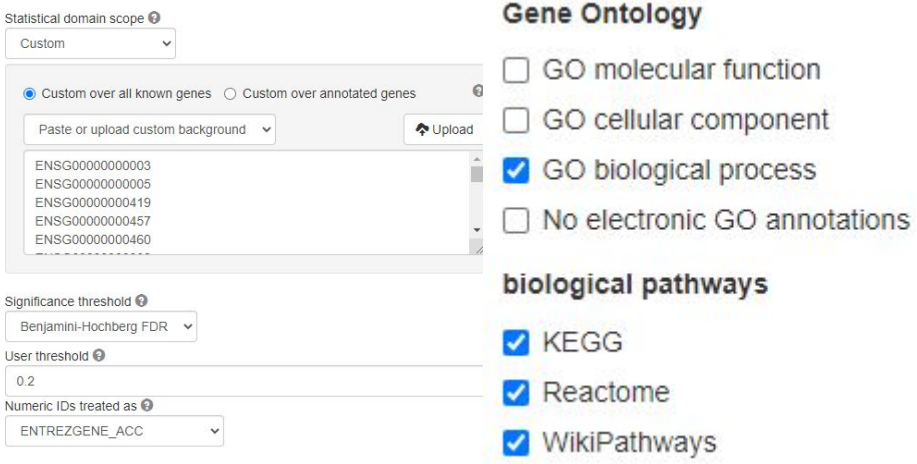
```
types = [LumA, LumB, her2, basal, normal]
names = ['LumA', 'LumB', 'her2', 'basal', 'normal']
for type, name in zip(types,names):
    type_data = list(type.Feature)
    type_input = [item.split('.', 1)[0] for item in type_data]
    gprofile_input = '\n'.join(type_input)
    text_file = open(rf'C:\Users\javte\Documents\SSEF project\datasets\gprofiler_inputs\redo_inputs\{name}.txt', "x")
    text_file.write(gprofile_input)
    text_file.close()
```

	Feature	Ranking
9720	ENSG00000160753.16	1
9052	ENSG00000154678.18	1
18005	ENSG00000230838.1	1
13022	ENSG00000178662.16	1
1369	ENSG00000078596.11	1
...	...	...
13739	ENSG00000183682.8	1
6992	ENSG00000137731.14	1
7332	ENSG00000139874.6	1
6511	ENSG00000135046.14	1
6226	ENSG00000132970.14	1

395 rows × 2 columns



gProfiler interface



Genes used for random selection: All genes in LumA dataset

p < 0.2

Only GO:BP, KEGG, REAC and WikiPathways were searched

Over-Representation Analysis

	source	term_name	term_id	adjusted_p_value	negative_log10_of_adjusted_p_value	term_size	query_size	intersection_size	effective_domain_size
0	GO:BP	multicellular organismal process	GO:0032501	0.000034	4.468801	6221	395	150	24833
1	GO:BP	anatomical structure development	GO:0048856	0.000058	4.237392	5237	395	130	24833
2	GO:BP	developmental process	GO:0032502	0.000343	3.465028	5712	395	135	24833
3	GO:BP	regulation of organelle organization	GO:0033043	0.000343	3.465028	1125	395	42	24833
4	GO:BP	cytoskeleton organization	GO:0007010	0.000368	3.433988	1407	395	48	24833

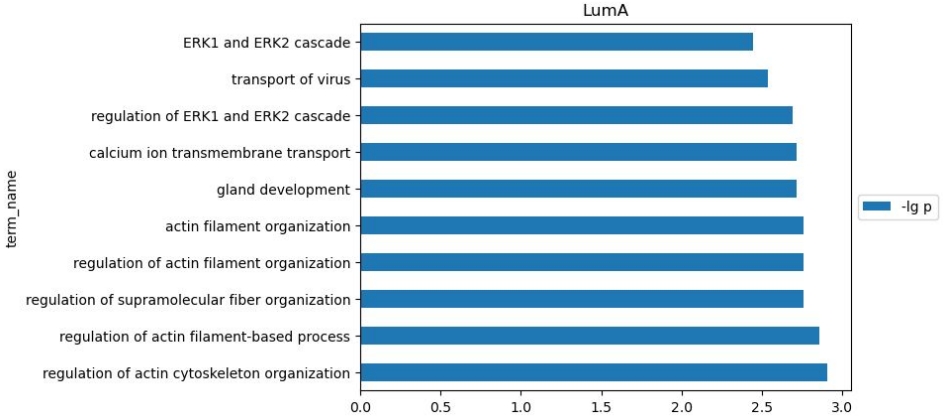
- p≤0.01
- source = GO:BP
- 15≤term size≤500

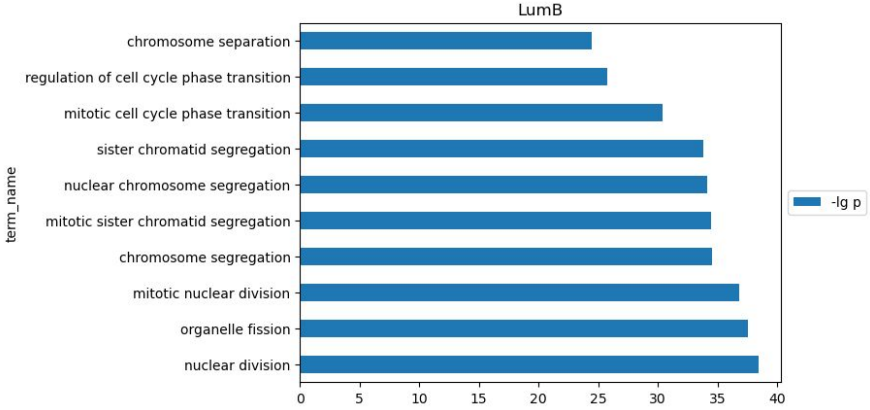
```
def filtered_pathways(gprofiler):
    gprofiler.query('adjusted_p_value <= 0.01 and source == "GO:BP" and 15<=term_size<=500',inplace=True)
    return gprofiler
```

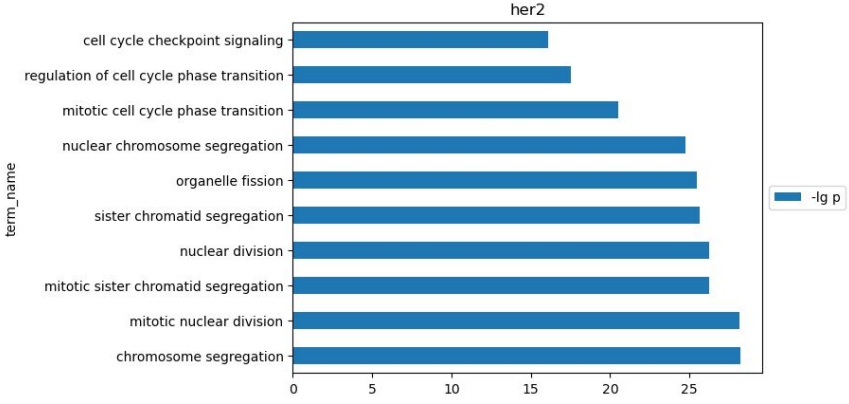
	source	term_name	term_id	adjusted_p_value	negative_log10_of_adjusted_p_value	term_size	query_size	intersection_size	effective_domain_size
11	GO:BP	regulation of actin cytoskeleton organization	GO:0032956	0.001239	2.906786	345	395	19	24833
14	GO:BP	regulation of actin filament-based process	GO:0032970	0.001402	2.853379	385	395	20	24833
15	GO:BP	regulation of supramolecular fiber organization	GO:1902903	0.001759	2.754693	365	395	19	24833
16	GO:BP	regulation of actin filament organization	GO:0110053	0.001759	2.754693	265	395	16	24833
18	GO:BP	actin filament organization	GO:0007015	0.001759	2.754693	433	395	21	24833
19	GO:BP	gland development	GO:0048732	0.001925	2.715498	404	395	20	24833

Over-Representation Analysis

- -lg p for top 10 GO:BP pathways was plotted for all molecular subtypes



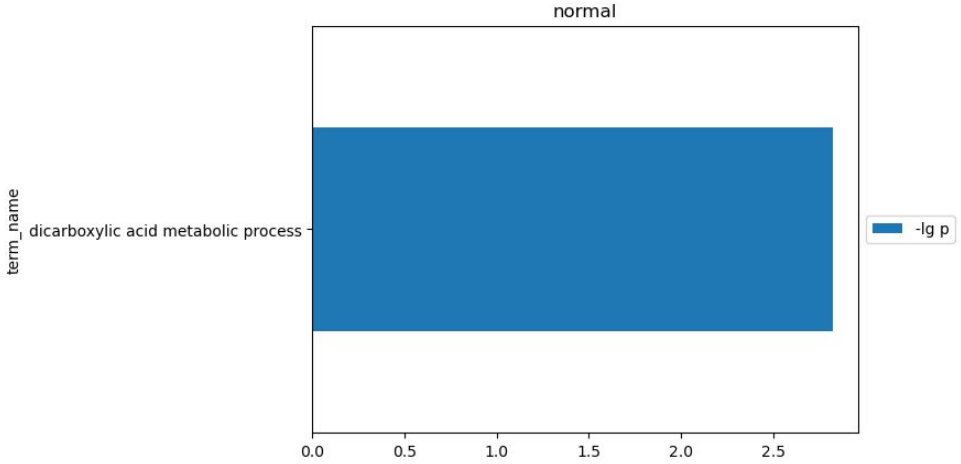


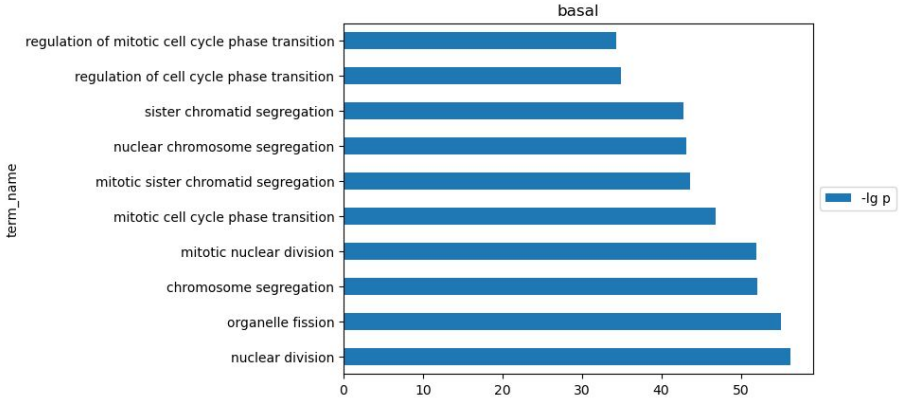


Mapping GO terms across molecular subtypes

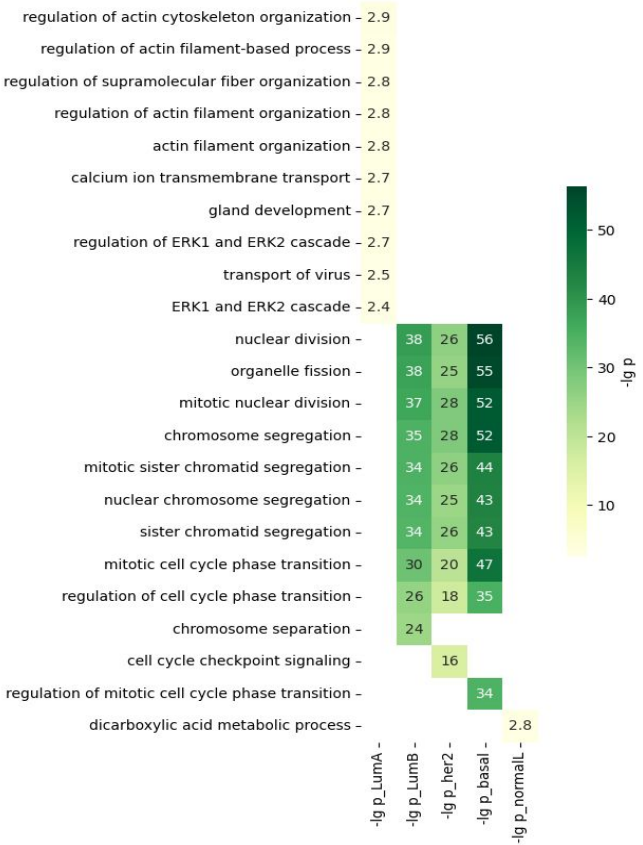
	LumA	LumB	Her2	basal	normal-Like
regulation of actin cytoskeleton organization					
regulation of ERK1 and ERK2 cascade					
transport of virus					
gland morphogenesis					
M Phase					
organelle fission					
mitotic nuclear division					
Cell Cycle Checkpoints					
mitotic cell cycle phase transition					
Mitotic Metaphase and Anaphase					
Mitotic Spindle Checkpoint					
dicarboxylic acid metabolic process					
PPAR signaling pathway					
Protein processing in endoplasmic reticulum					
Asparagine N-linked glycosylation					
COPI-dependent Golgi-to-ER retrograde traffic					
Golgi-to-ER retrograde transport					

Over-Representation Analysis



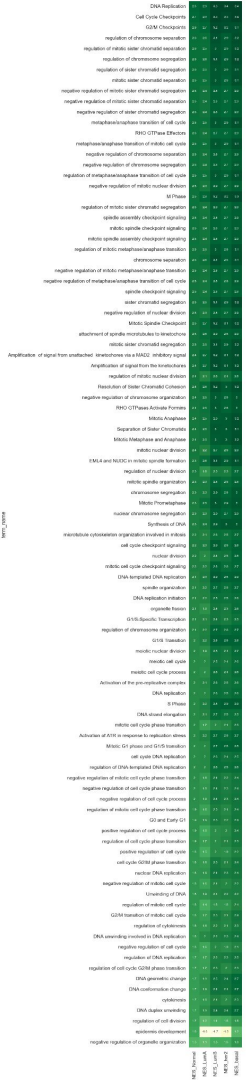


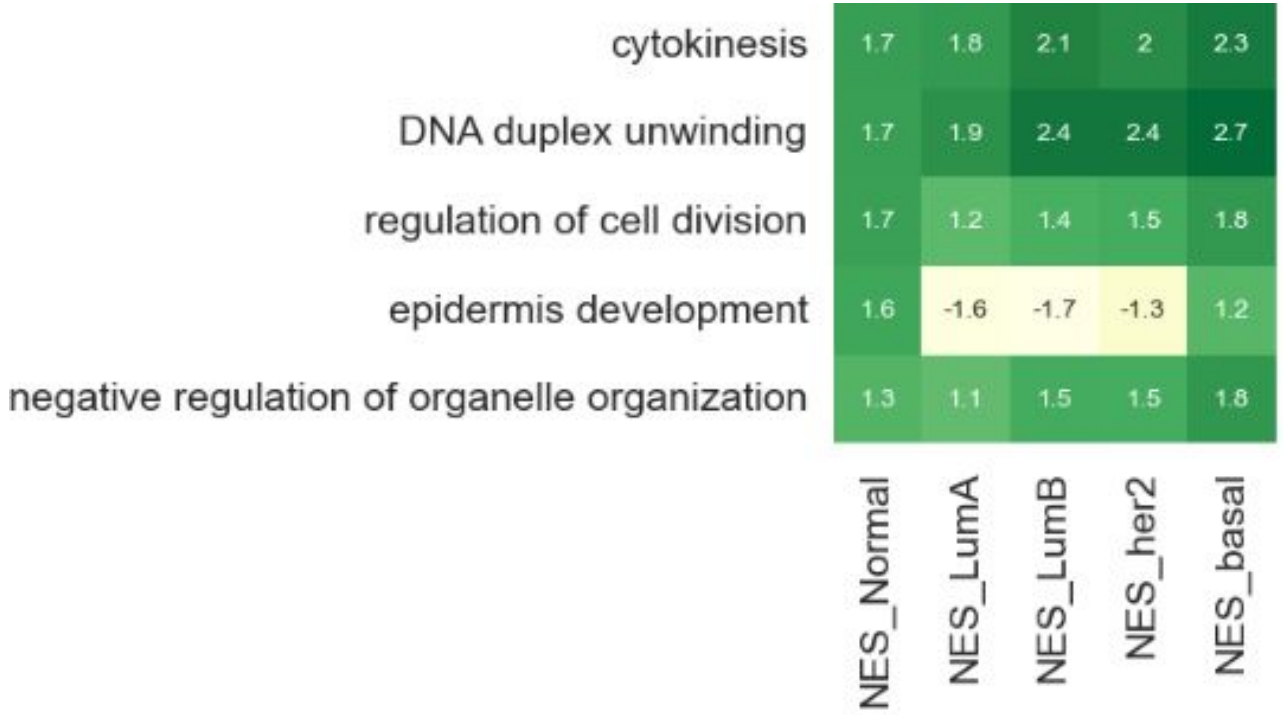
Mapping GO terms for across molecular subtypes by top 10 hits

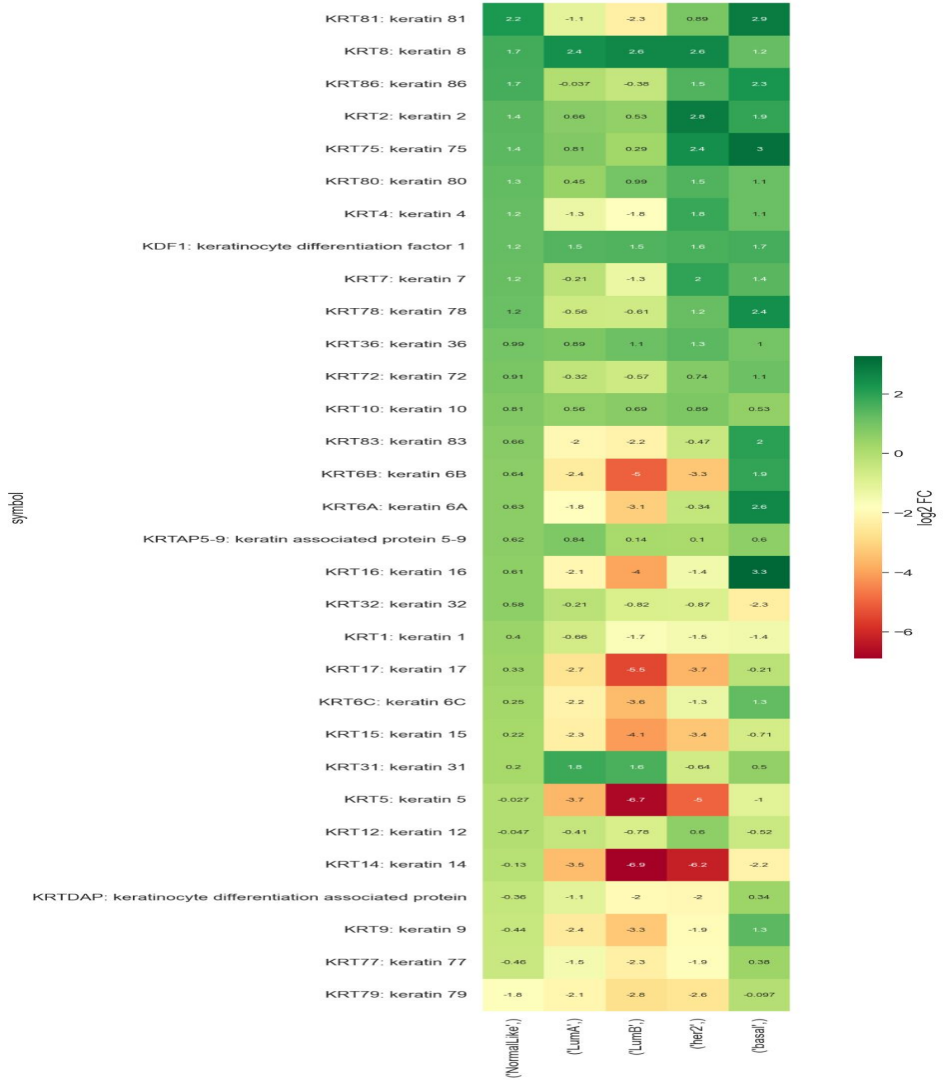


		regulation of actin filament organization		
	mapping GO terms for across molecular subtypes	calcium ion transmembrane transport		
		gland development		
		regulation of ERK1 and ERK2 cascade		
		transport of virus		
		ERK1 and ERK2 cascade		
		gland morphogenesis		
		M Phase		
		nuclear division		
		organelle fission		
		mitotic nuclear division		
		chromosome segregation		
		mitotic sister chromatid segregation		
		nuclear chromosome segregation		
		sister chromatid segregation		
		Cell Cycle Checkpoints		
		mitotic cell cycle phase transition		
		Mitotic Metaphase and Anaphase		
		Mitotic Anaphase		
		Mitotic Spindle Checkpoint		

Selected enriched pathways





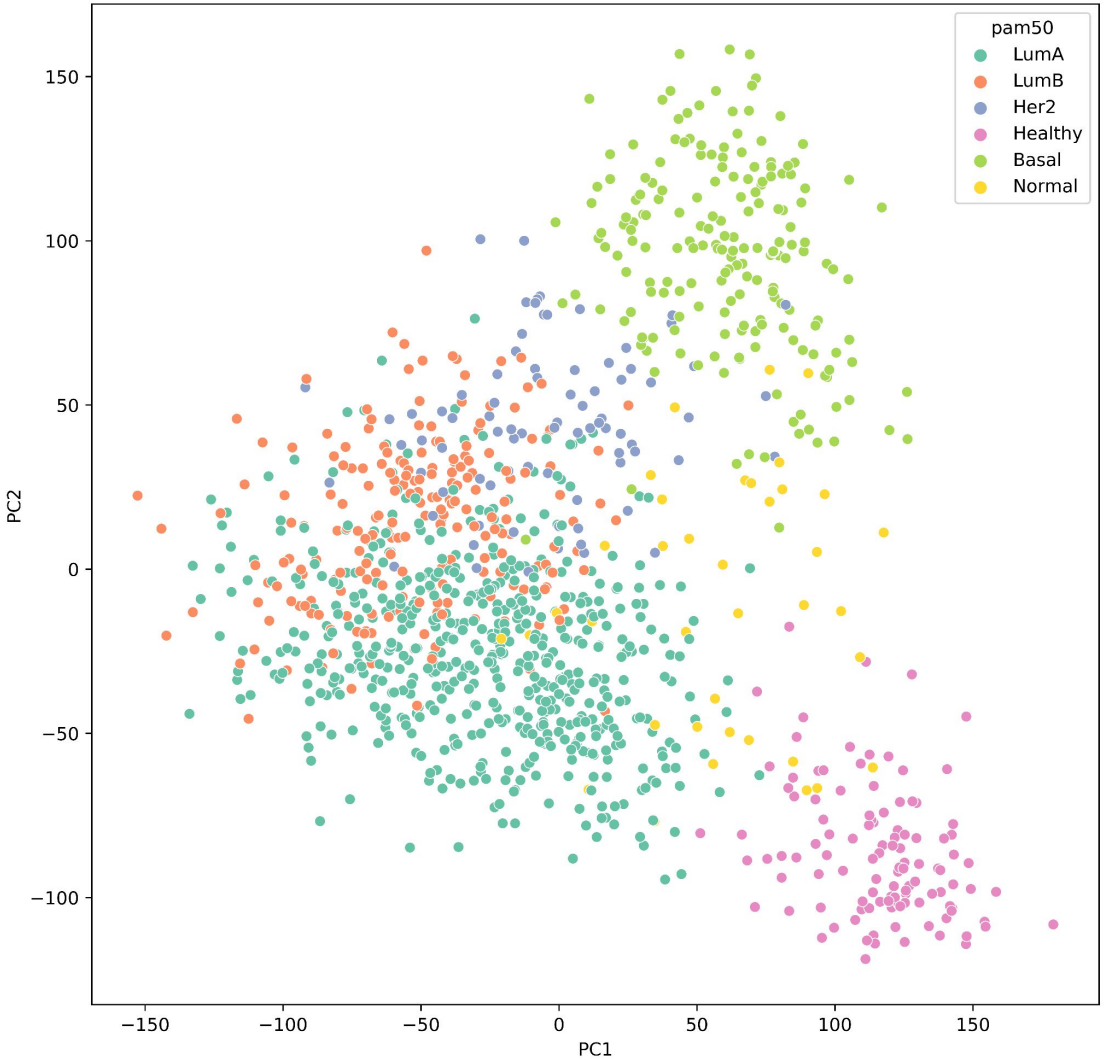


in epidermis development

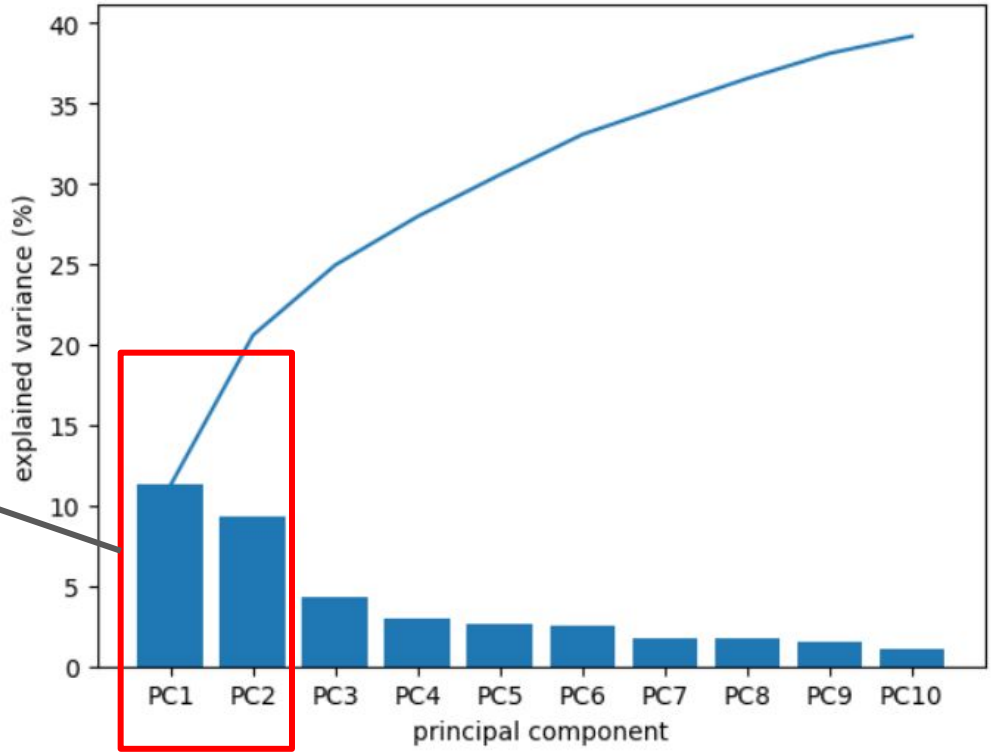
Hypothesis: DE genes in immune cells will differentiate across tumor subtypes

PCA for all subtypes

- k-means clustering was performed on PCA projection



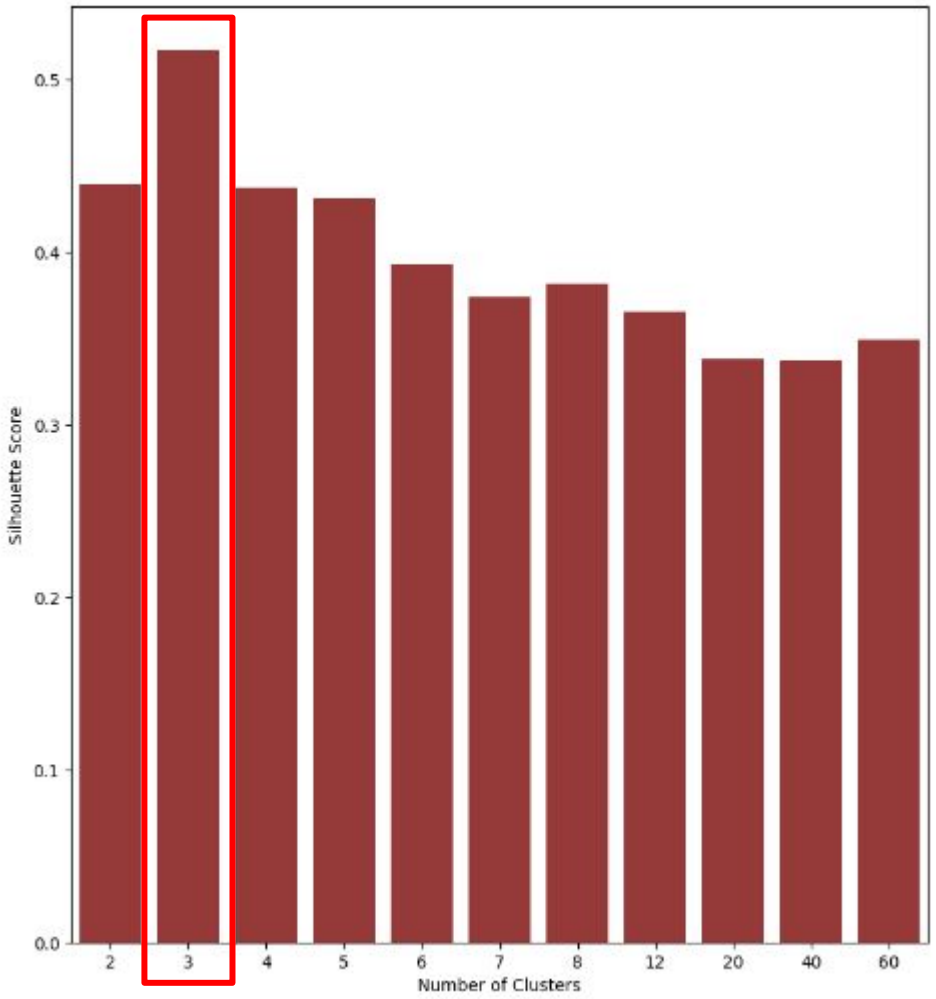
20.6% of variance



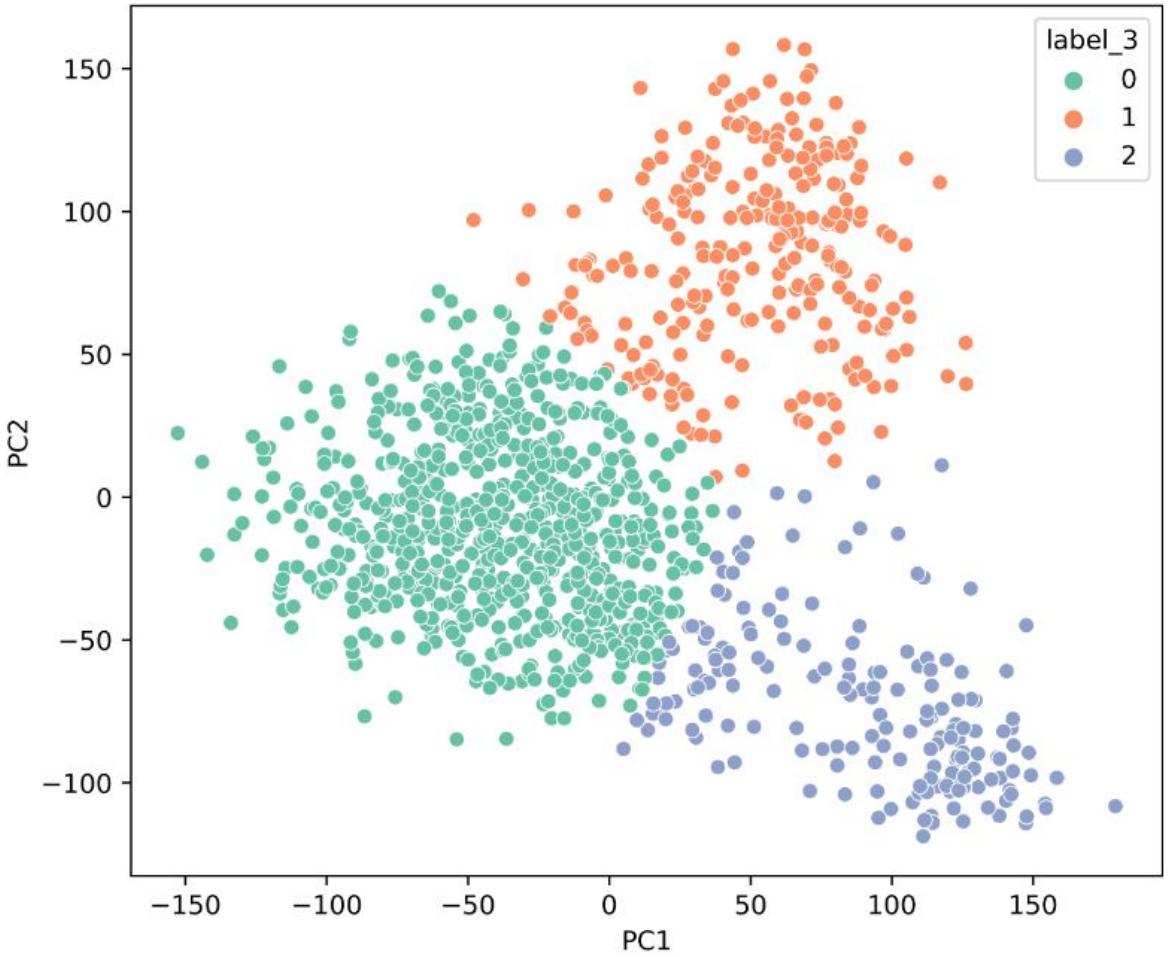
- The silhouette value measures how similar a point is to its own cluster compared to other clusters

- Silhouette score = mean silhouette value over all data of the entire dataset

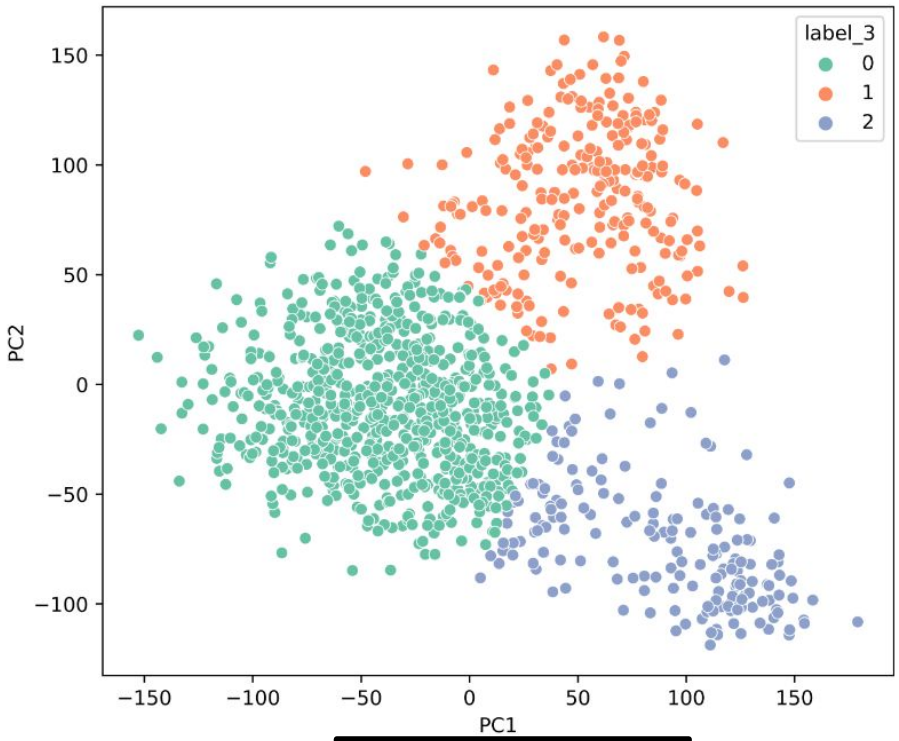
- Plot of silhouette score vs number of clusters suggests that k = 3 is optimal



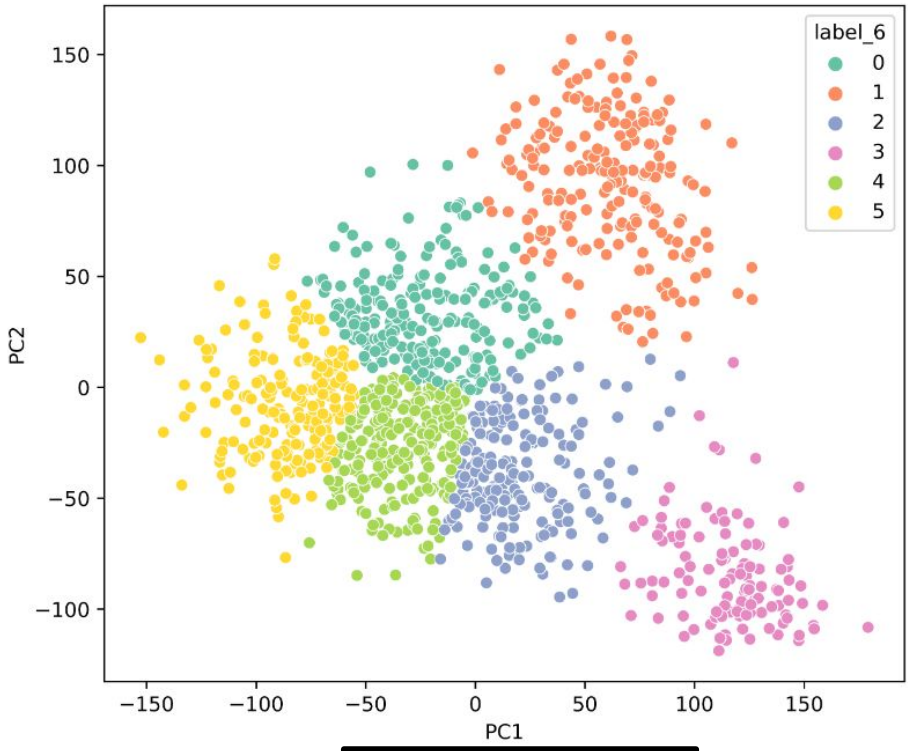
k-means clustering
for k = 3



- Thus, k = 6 was used to see if samples could be classified correctly

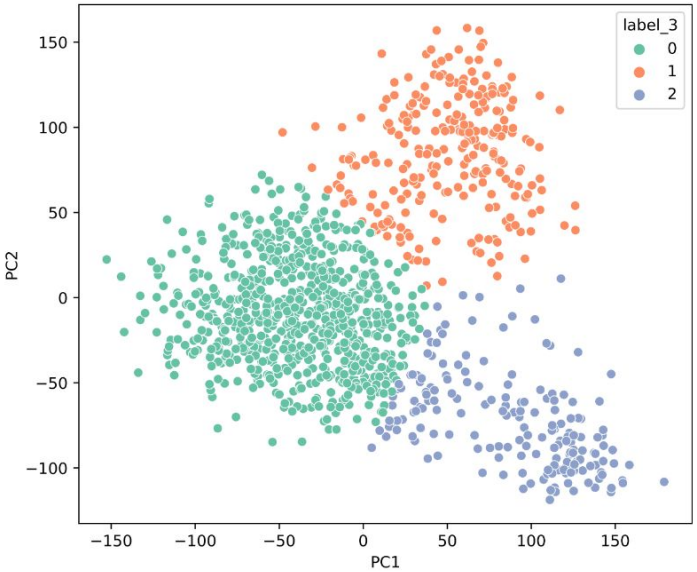


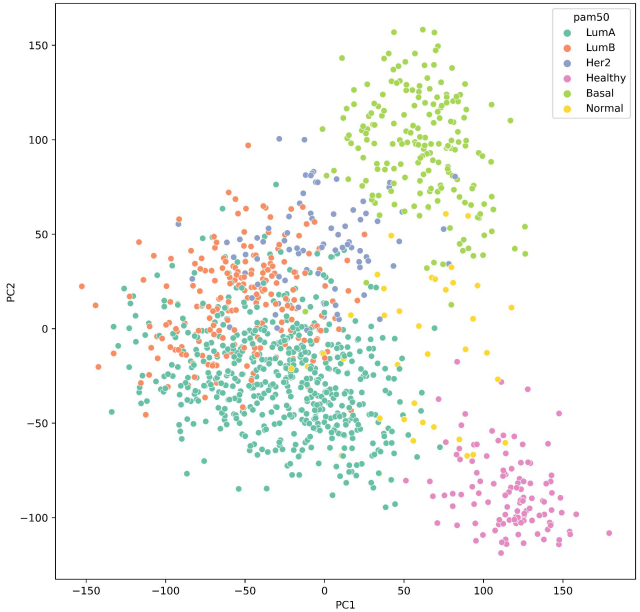
k = 3



k = 6

Evaluation of k-means clustering

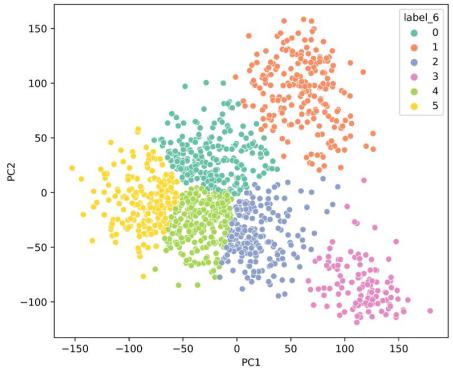


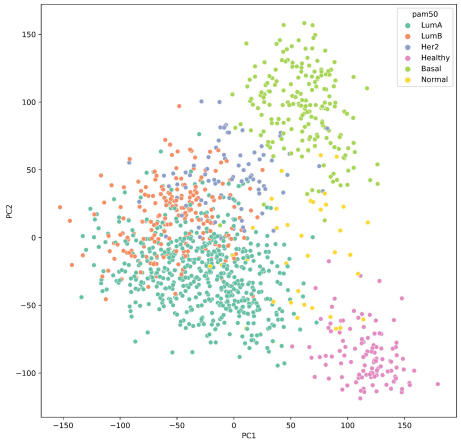


- Using 3 clusters merely separates basal and healthy samples from the rest

Evaluation of k-means clustering for k = 6

- Comparison of assigned labels by k-means clustering to actual pam50 label by using a confusion matrix
- Similar to k = 3, only basal and healthy samples were able to be separated cleanly from the rest
- Is confused between LumA, LumB and Her2





Hierarchical clustering

- Hierarchical clustering was performed on the original dataset

```
#hierarchal clustering
from scipy.cluster.hierarchy import fcluster, linkage
distance_matrix = linkage(full_matrix, method = 'ward', metric = 'euclidean')
```

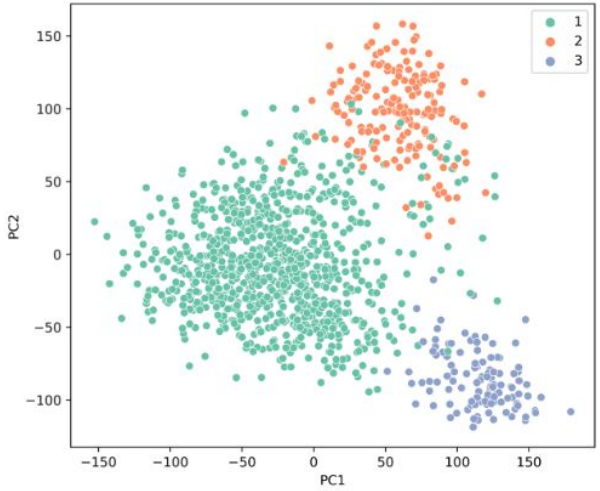
- Ward method and Euclidean distance metric was used

```
hierarchal_labels_3 = fcluster(distance_matrix, 3, criterion='maxclust')
hierarchal_labels_6 = fcluster(distance_matrix, 6, criterion='maxclust')
```

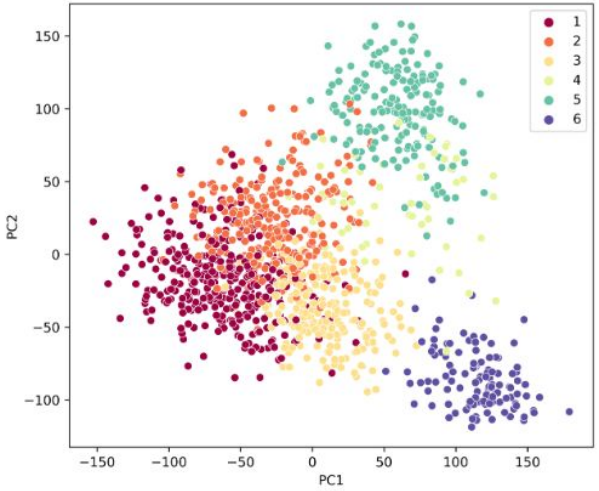
- Performed hierarchical clustering to get 3 and 6 clusters with maxclust criterion

- Hierarchical clustering of the samples was then visualised in PC space

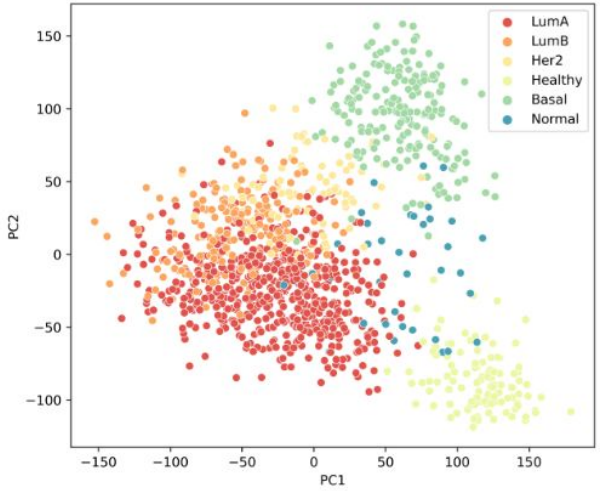
Hierarchical clustering



h_cluster = 3



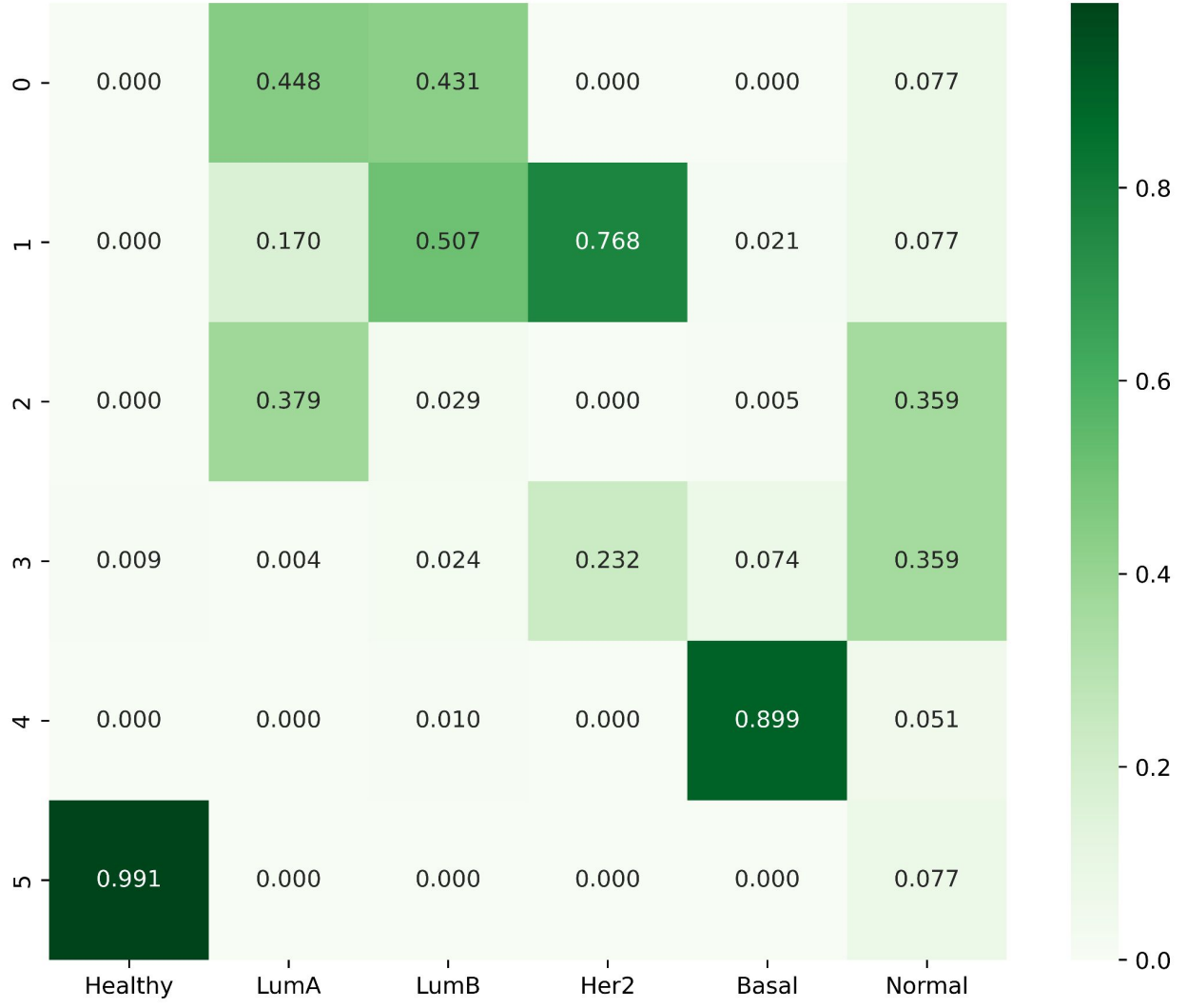
h_cluster = 6



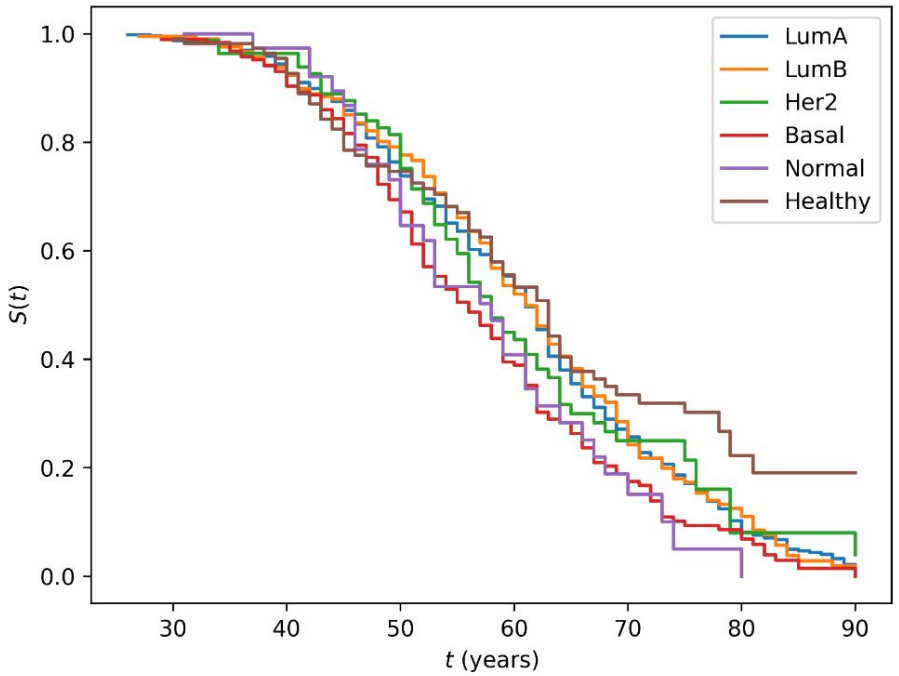
Original labels

Evaluation of hierarchical clustering

- Classified healthy more accurately than k-means
- Still intermingling between LumA, LumB and Her2



Kaplan - Meier Survival Analysis

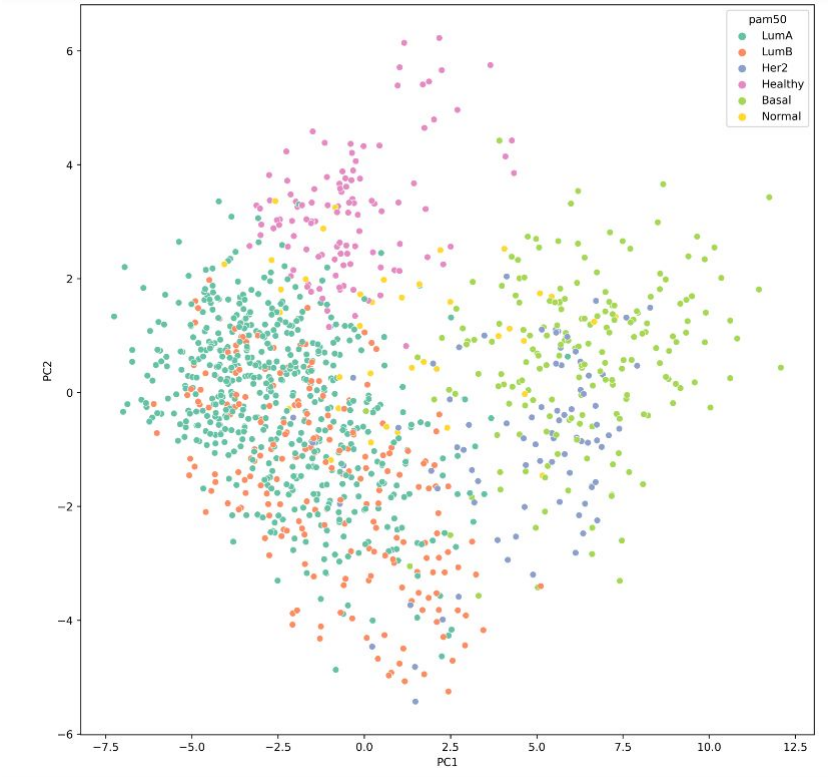


- Basal and Normal like have the lowest probability of survival

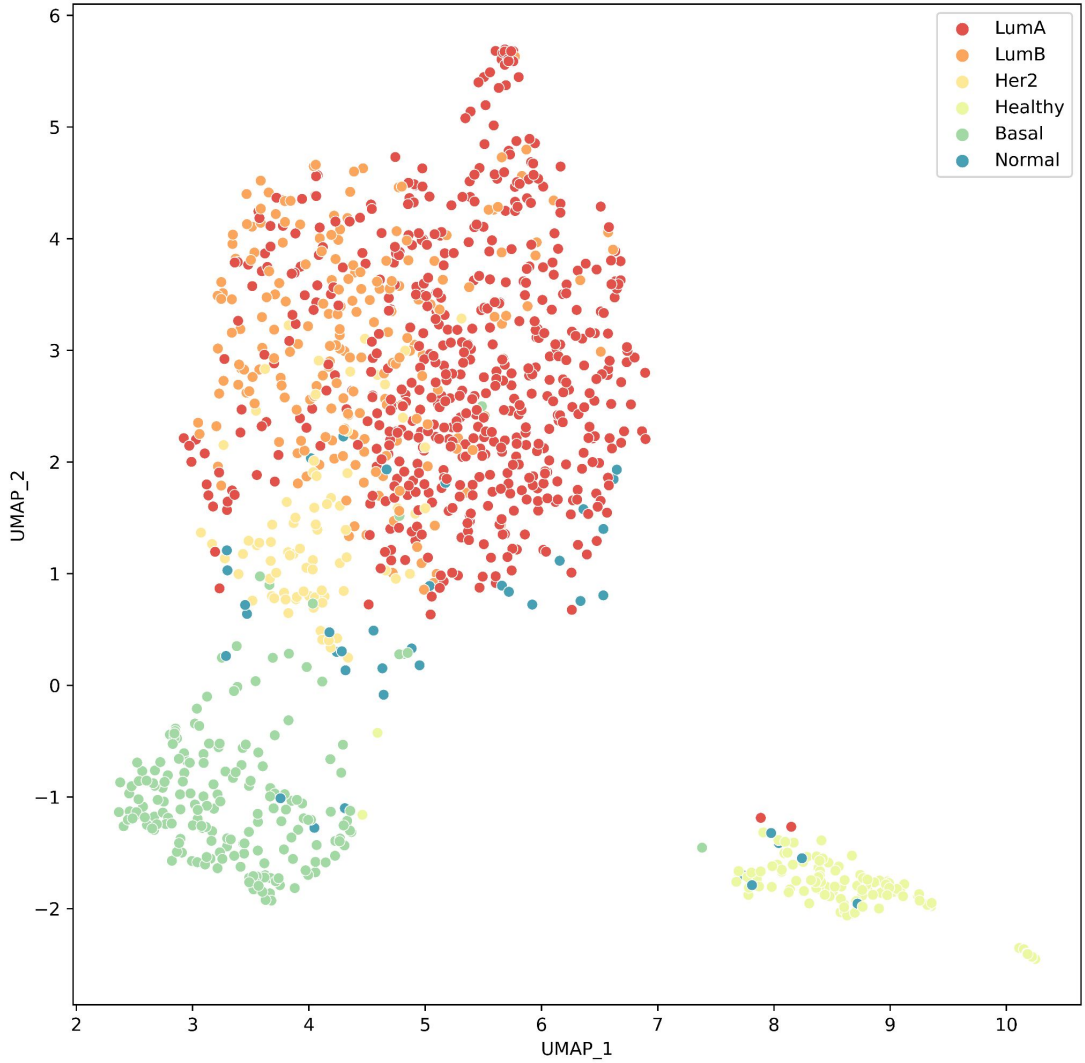
Using only ER, PR, HER2, and Ki67 genes for PCA

	ERBB2	ESR1	ESR2	PGR	MKI67
TCGA-AC-A8OP-01A-11R-A36F-07	2.432012	10.823444	-1.622318	9.162394	6.095878
TCGA-D8-A1XU-01A-11R-A14M-07	5.340929	10.154951	-1.113781	7.588521	4.702064
TCGA-BH-A18L-01A-32R-A12D-07	5.960115	9.881238	-1.222892	8.929414	8.074962
TCGA-B6-A0IK-01A-12R-A056-07	3.709075	5.400489	-0.265877	8.598221	6.342301
TCGA-BH-A18L-11A-42R-A12D-07	6.557885	8.198074	1.895983	7.156401	2.328696
...	...	...	...	...	...
TCGA-GM-A2DN-01A-11R-A180-07	4.864716	8.136211	-0.587039	8.283925	6.412742
TCGA-B6-A0I1-01A-11R-A21T-07	-2.132815	1.127745	-1.859734	8.356803	6.259360
TCGA-EW-A1IW-01A-11R-A13Q-07	7.535994	8.548899	-1.187212	12.384211	6.697537
TCGA-BH-A42V-01A-11R-A24H-07	5.675222	5.941458	0.181847	8.849644	5.126441
TCGA-AR-A251-01A-12R-A169-07	-0.258195	4.172118	0.313161	7.258320	7.690807

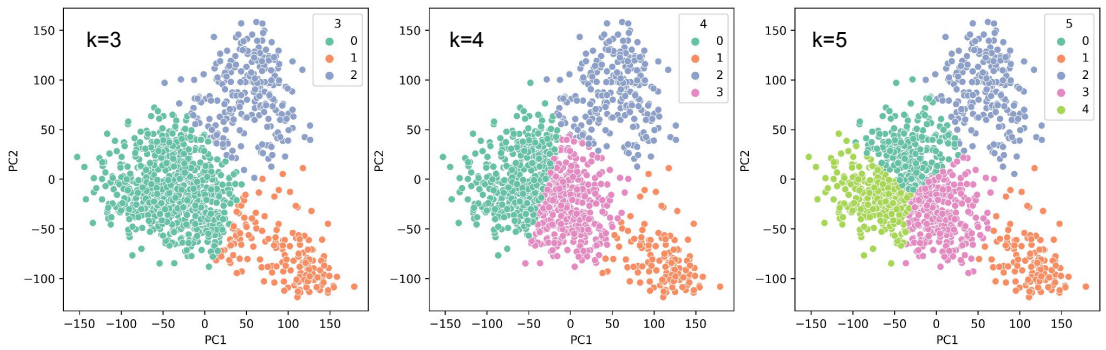
- Not much better at classifying compared to using all 26000 genes



UMAP projection of original dataset

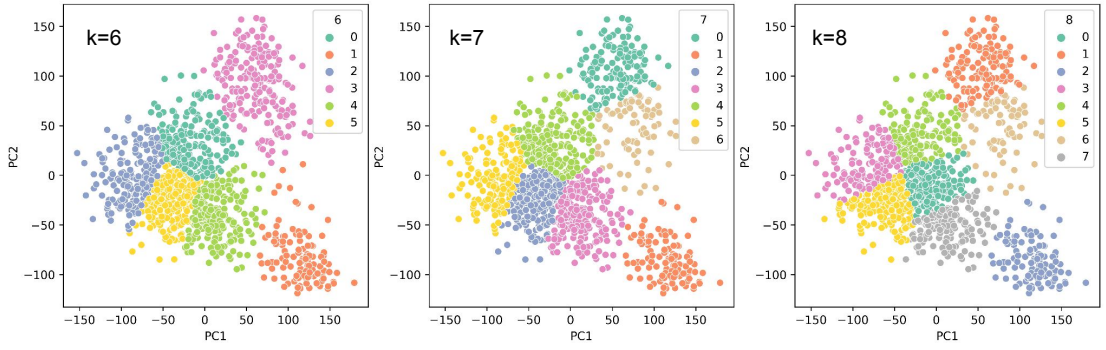


Clustering results obtained using k-means algorithm with various values of k


k=3
3
0
1
2
PC1
PC2

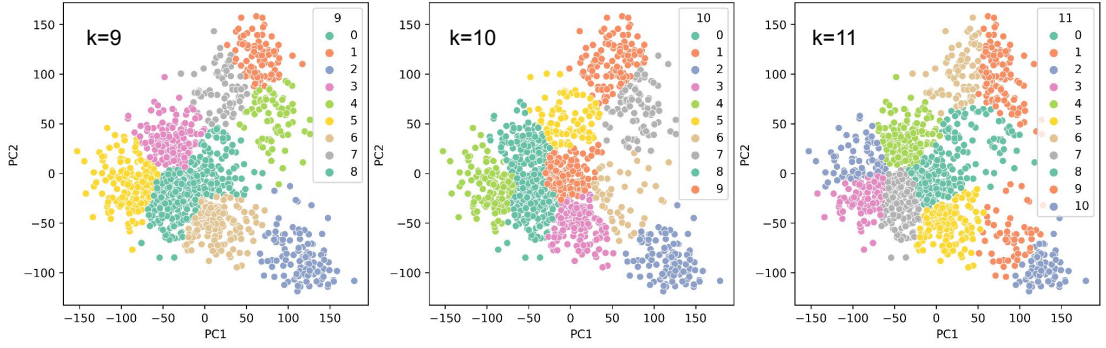
k=4
4
0
1
2
3
PC1
PC2

k=5
5
0
1
2
3
4
PC1
PC2


k=6
6
0
1
2
3
4
5
PC1
PC2

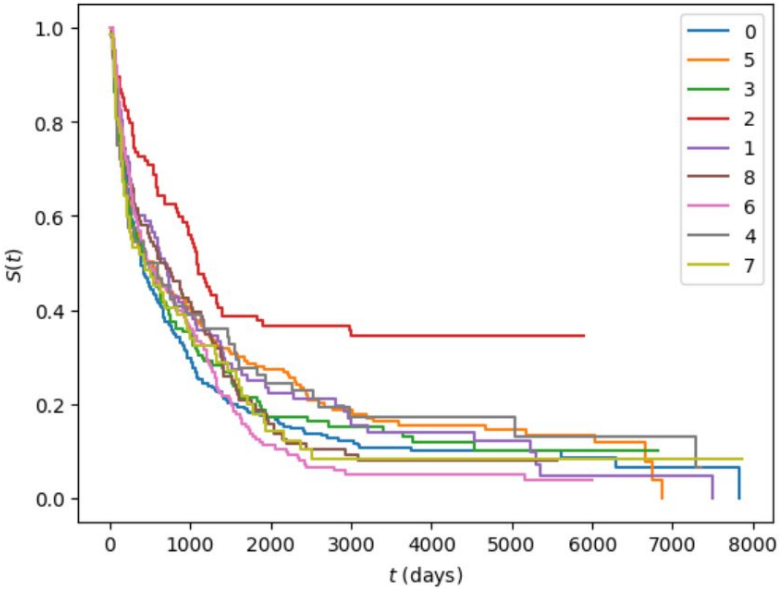
k=7
7
0
1
2
3
4
5
6
PC1
PC2

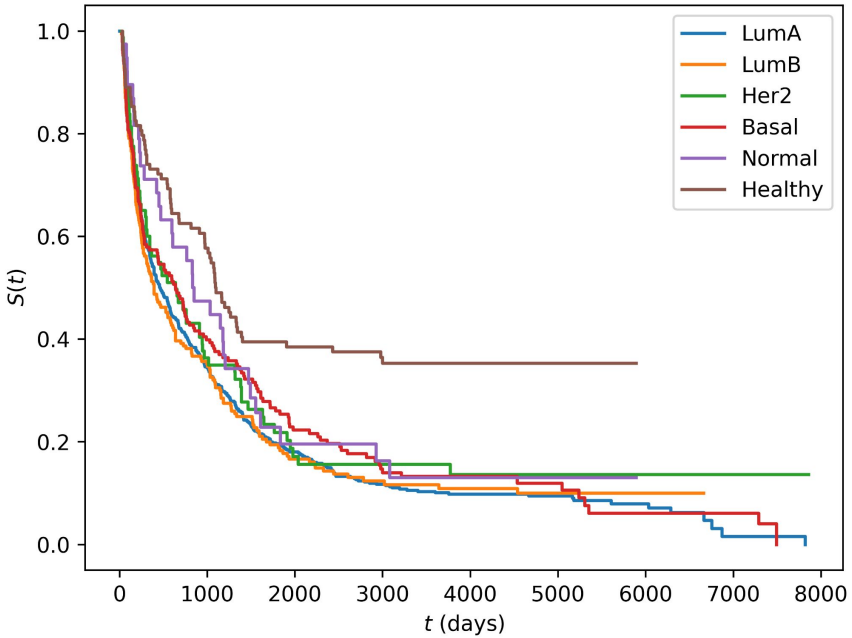
k=8
8
0
1
2
3
4
5
6
7
PC1
PC2

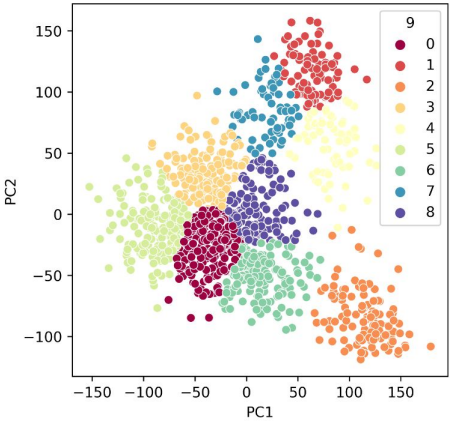

k=9
9
0
1
2
3
4
5
6
7
8
PC1
PC2

k=10
10
0
1
2
3
4
5
6
7
8
9
PC1
PC2

k=11
11
0
1
2
3
4
5
6
7
8
9
10
PC1
PC2


S(t)
t (days)
0
5
3
2
1
8
6
4
7


S(t)
t (days)
LumA
LumB
Her2
Basal
Normal
Healthy


PC1
PC2
9
0
1
2
3
4
5
6
7
8

Ensemble of clustering results

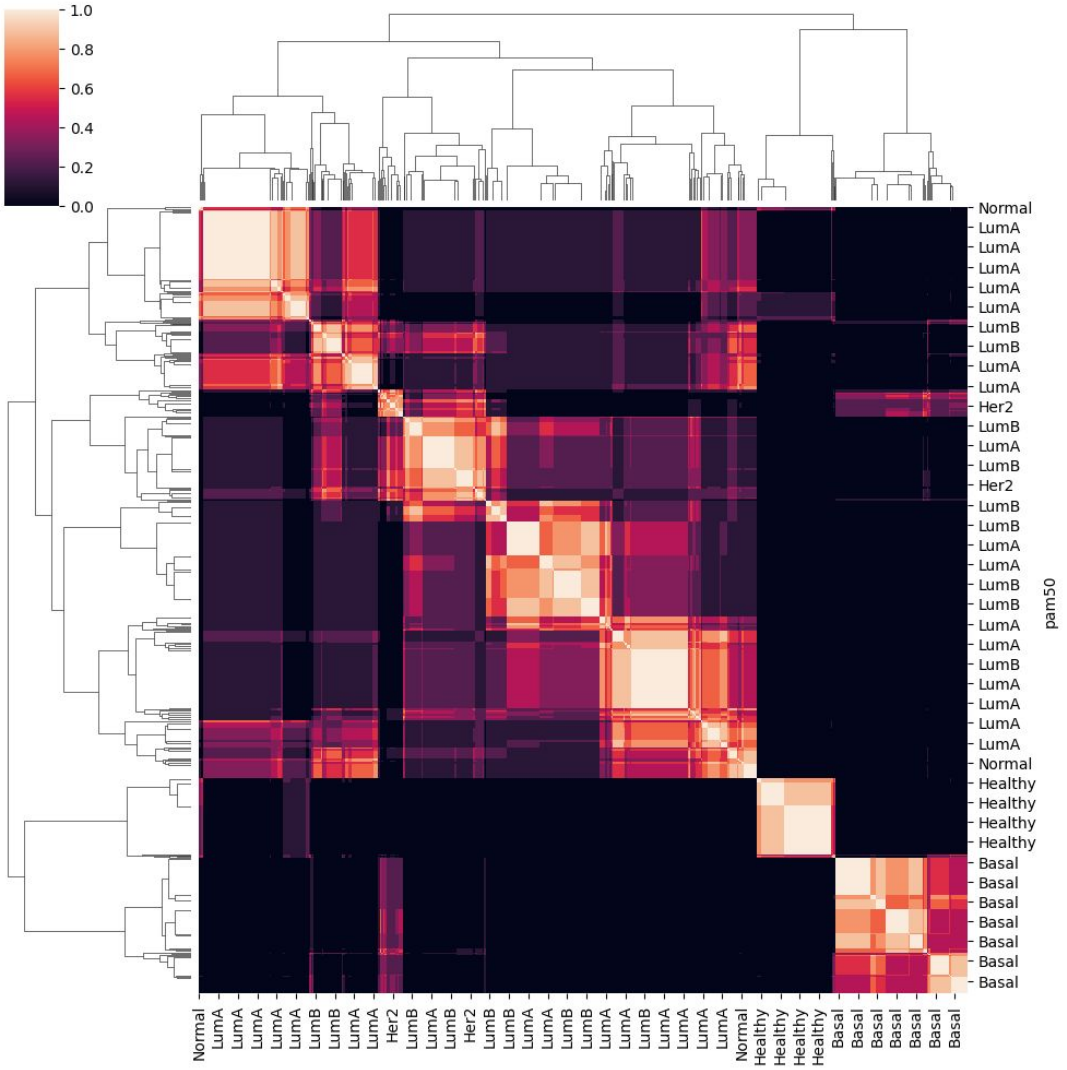
sample	k= 3	4	5	6	7	8	9	10	11
0	0	0	4	5	2	5	0	0	7
1	0	3	3	5	2	5	0	0	7

Sample 0 is clustered with Sample 1 7 times out of 9

Entry (0,1) in co-occurrence matrix = 7/9 = 0.778

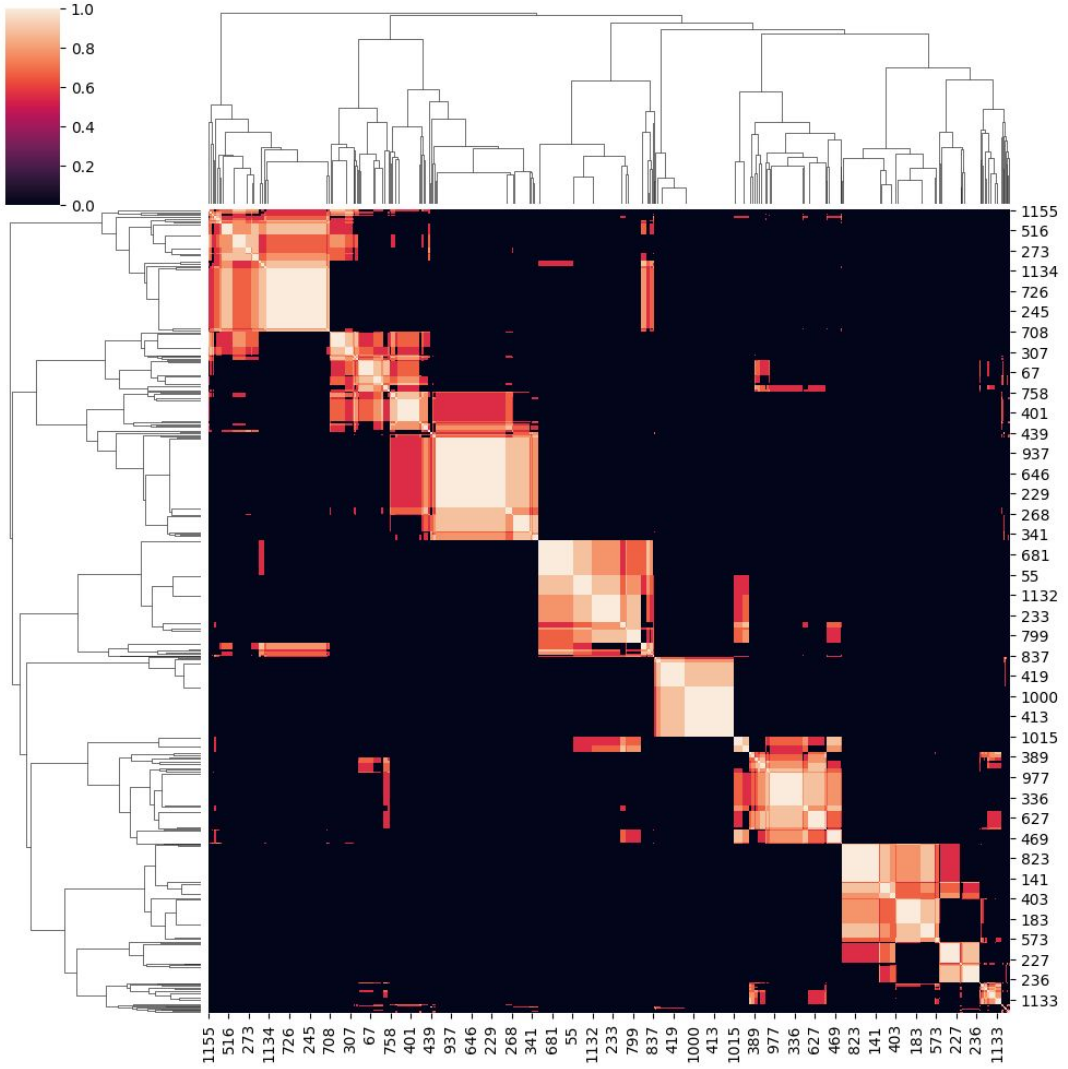
	0	1
0	1.000000	0.777778
1	0.777778	1.000000

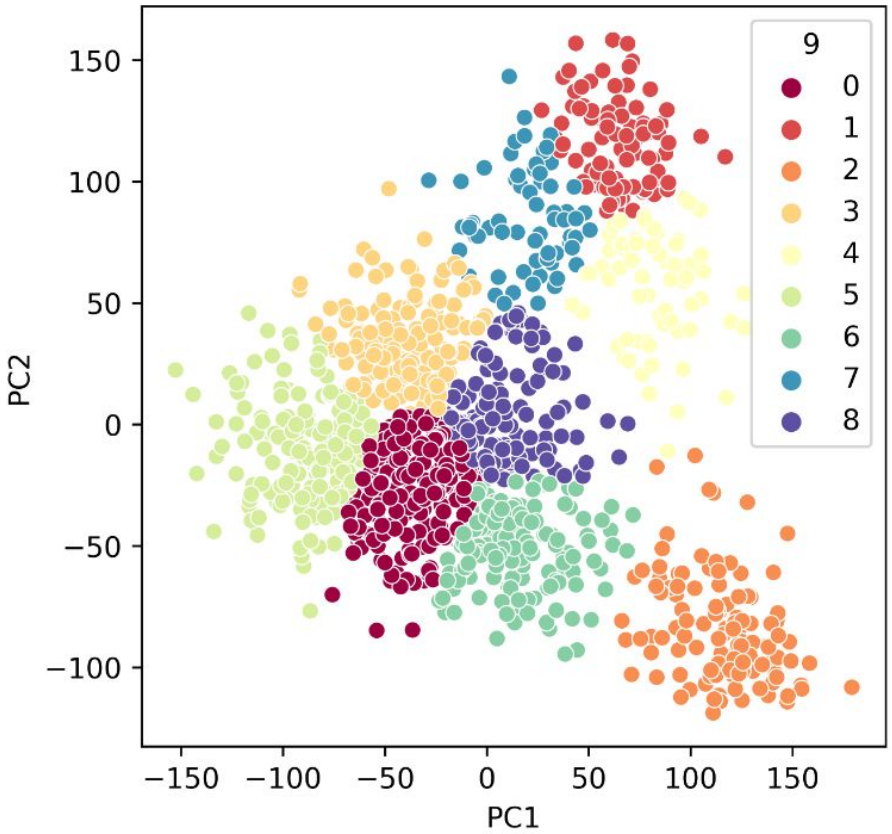
Co-occurrence matrix

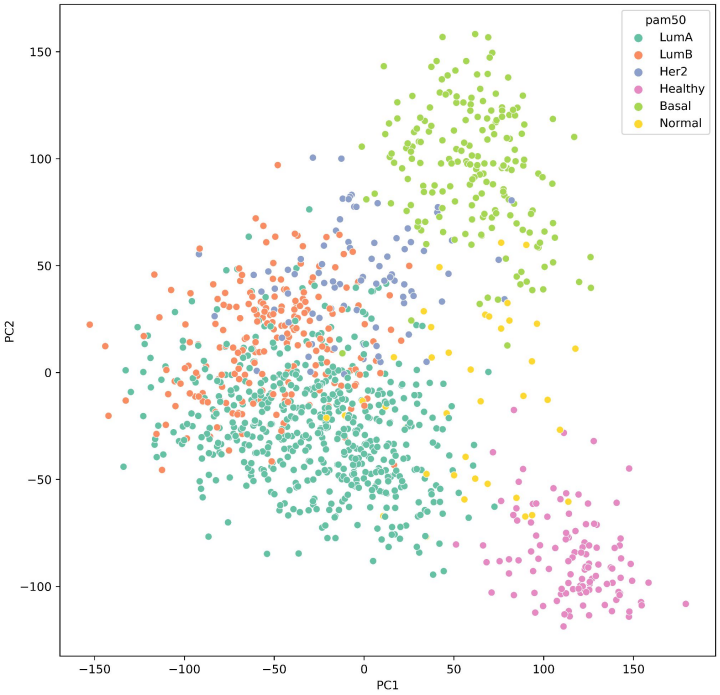


Cutoff set at 0.5

- 9 clusters can be observed




9
0
1
2
3
4
5
6
7
8
150
100
50
0
−50
−100
PC2
−150
−100
−50
0
50
100
150
PC1


pam50
LumA
LumB
Her2
Healthy
Basal
Normal
150
100
50
0
−50
−100
PC2
−150
−100
−50
0
50
100
150
PC1