EpiMed Open Course – Session 4

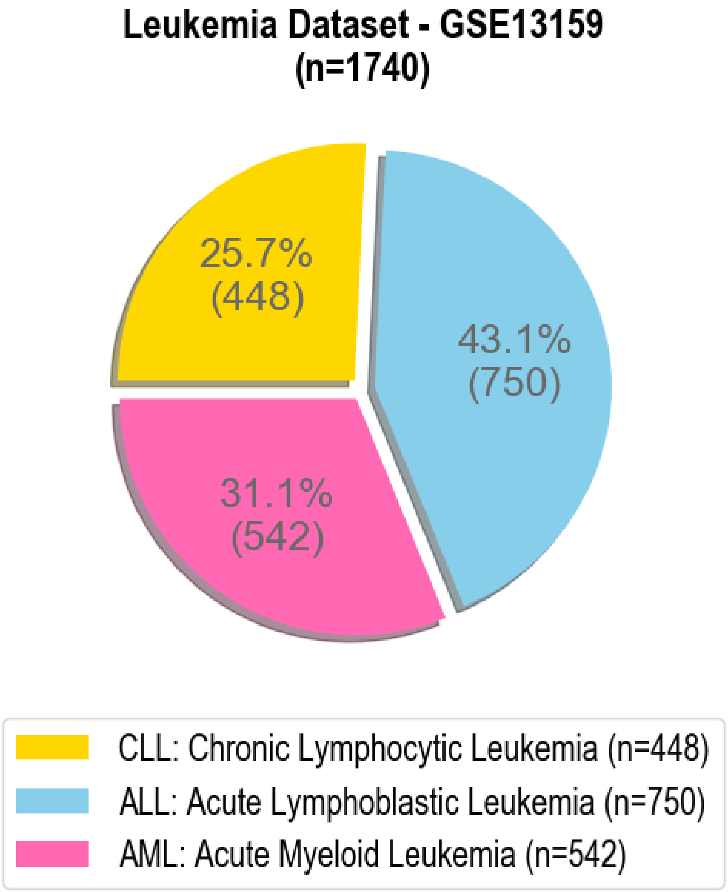
AI for Omics – Use case of leukemia classification – Part 2

Ekaterina Flin

13/04/2020

Use case: leukemia dataset

Goal: Predict leukemia type CLL, ALL or AML



Transcriptomic data, microarrays

Total number of samples = 1 740

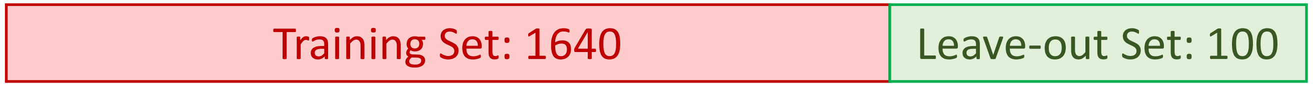
Total number of genes (“features”) = 21 875

3 labels to identify : CLL, ALL or AML

Pipeline for data preparation

Pipeline for data preparation

1 Split dataset



Training Set: 1640

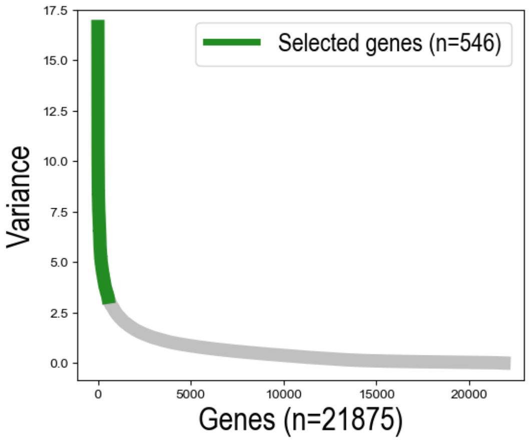
Leave-out Set: 100

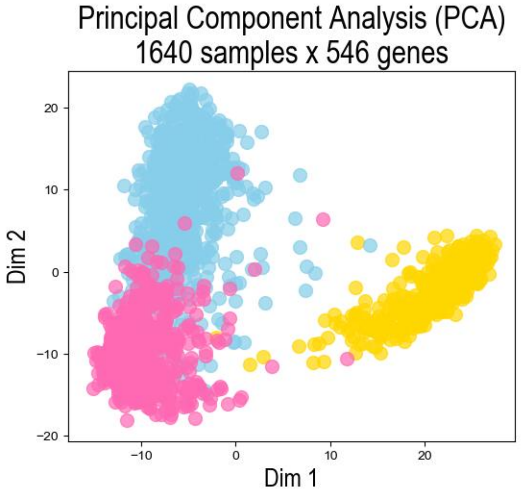
Pipeline for data preparation

1 Split dataset

Training Set: 1640	Leave-out Set: 100

2 Selection of variables



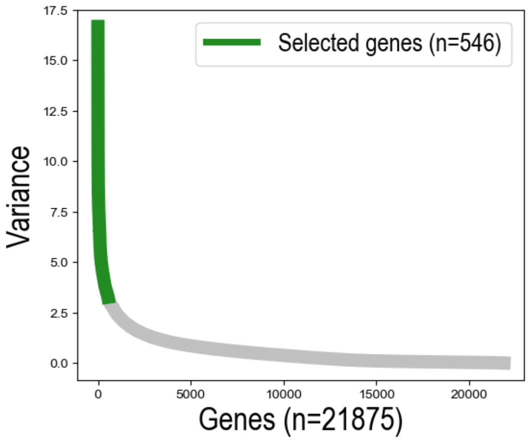

Principal Component Analysis (PCA)
1640 samples x 546 genes

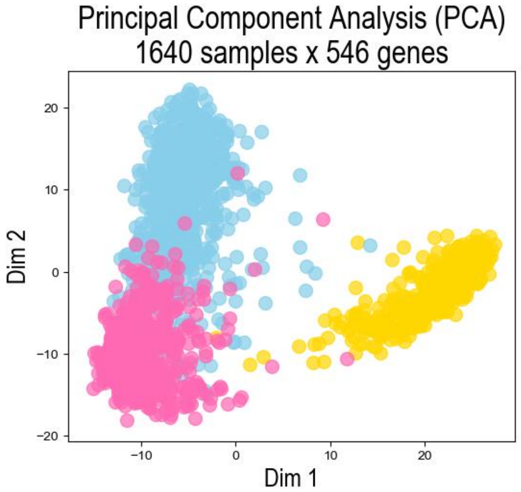
Pipeline for data preparation

1 Split dataset

Training Set: 1640	Leave-out Set: 100

2 Selection of variables





3 Cross-validation

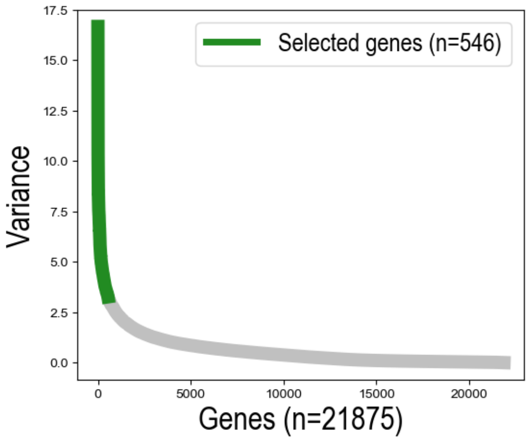
Train	Train	Test
Test	Train	Train
Train	Test	Train

Pipeline for data preparation

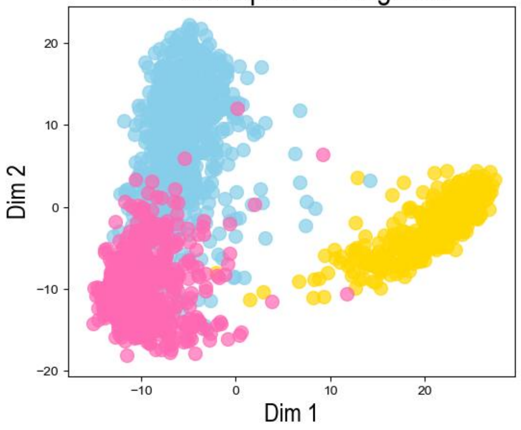
1 Split dataset

Training Set: 1640	Leave-out Set: 100

2 Selection of variables



Principal Component Analysis (PCA)
1640 samples x 546 genes



3 Cross-validation

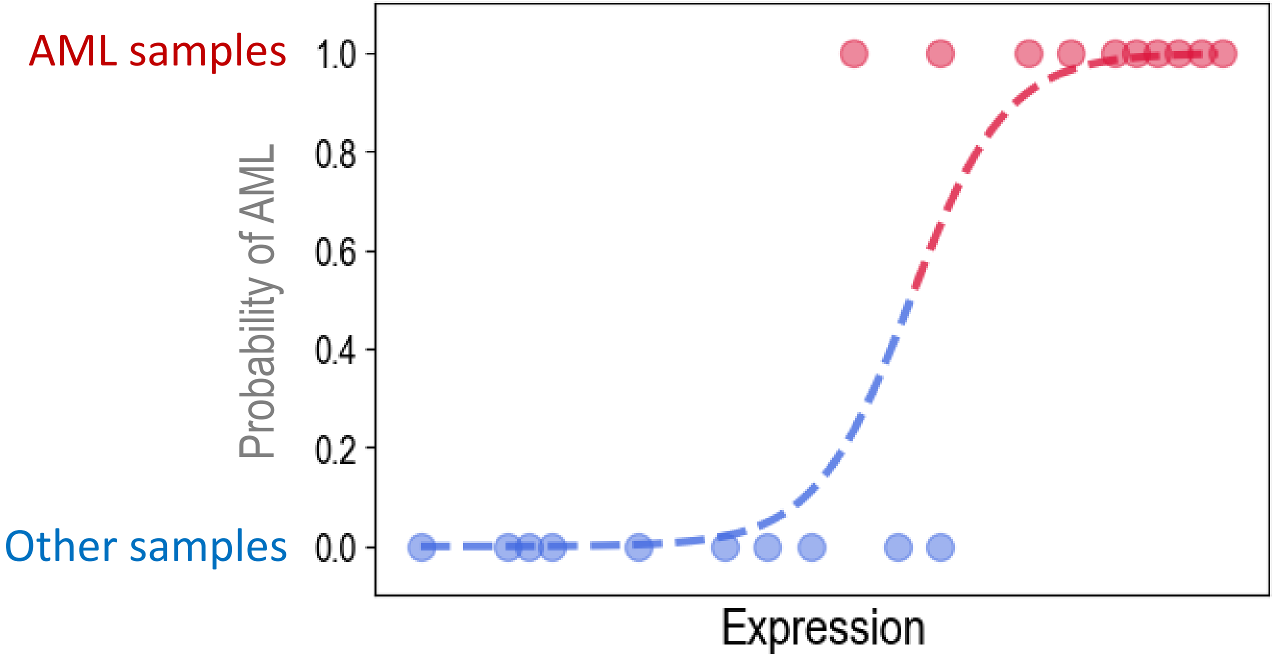
Train	Train	Test
Test	Train	Train
Train	Test	Train

4 Data scaling

Train	Train	Test

$$scaled\_expression = \frac{expression - \mu}{\sigma}$$

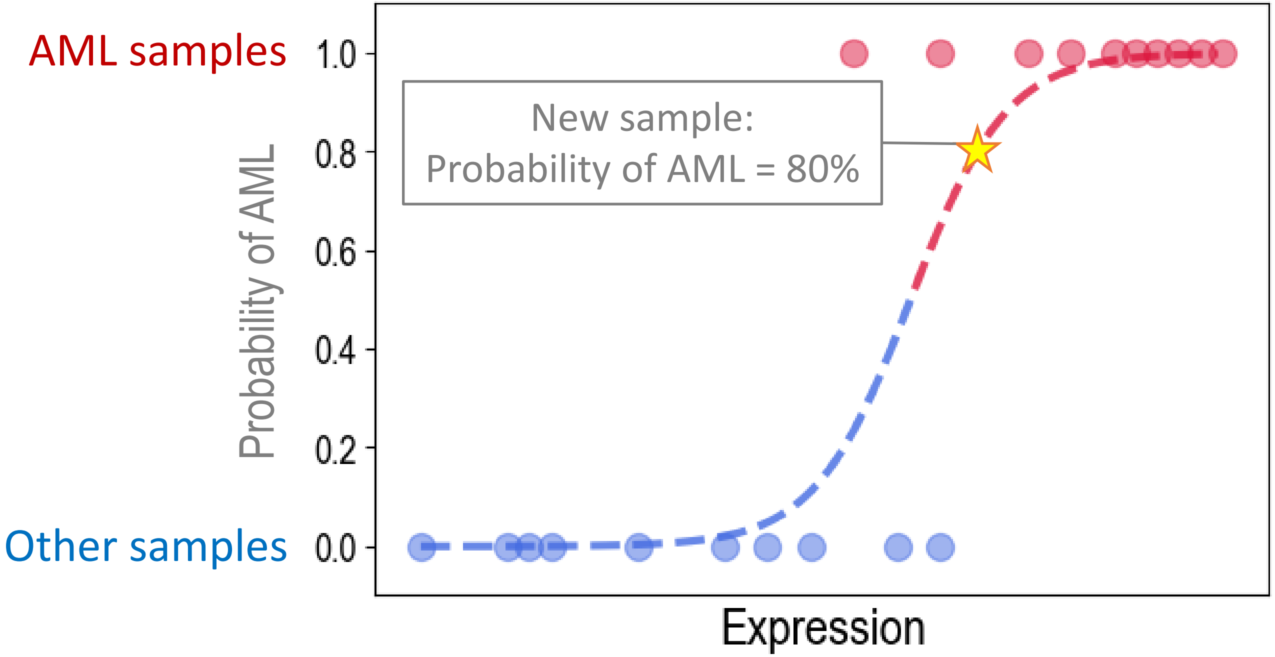
Train a classifier: Logistic Regression



Logistic function $$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^{x}}{1 + e^{x}}$$

S-shaped curve, also called “sigmoid” function

Train a classifier: Logistic Regression



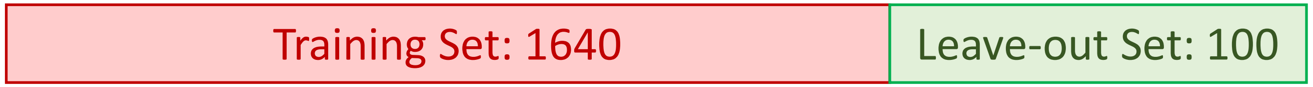
Logistic function $$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{1 + e^x}$$

S-shaped curve, also called “sigmoid” function

Logistic Regression in Python

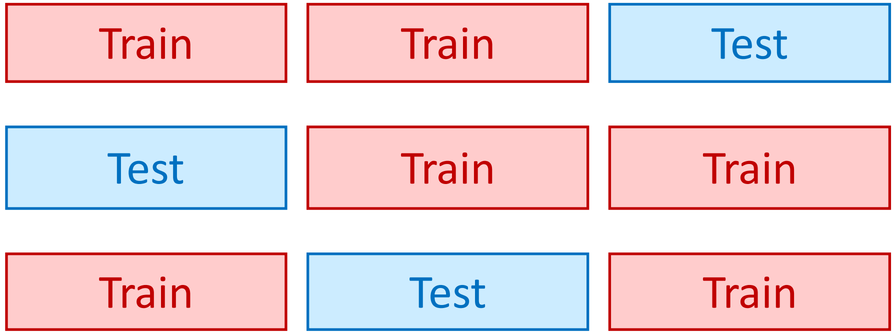
```python
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

# Create a classifier
classifier = LogisticRegression(multi_class='multinomial',
                          penalty='none', solver='newton-cg')

# Train classifier using training dataset
classifier.fit(X_train_scaled, y_train)

# Predict labels for test dataset
y_pred_test = classifier.predict(X_test_scaled)

# Calculate accuracy comparing prediction
# with real labels in test dataset
accuracy = metrics.accuracy_score(y_test, y_pred_test)
```

Train a classifier: Logistic Regression

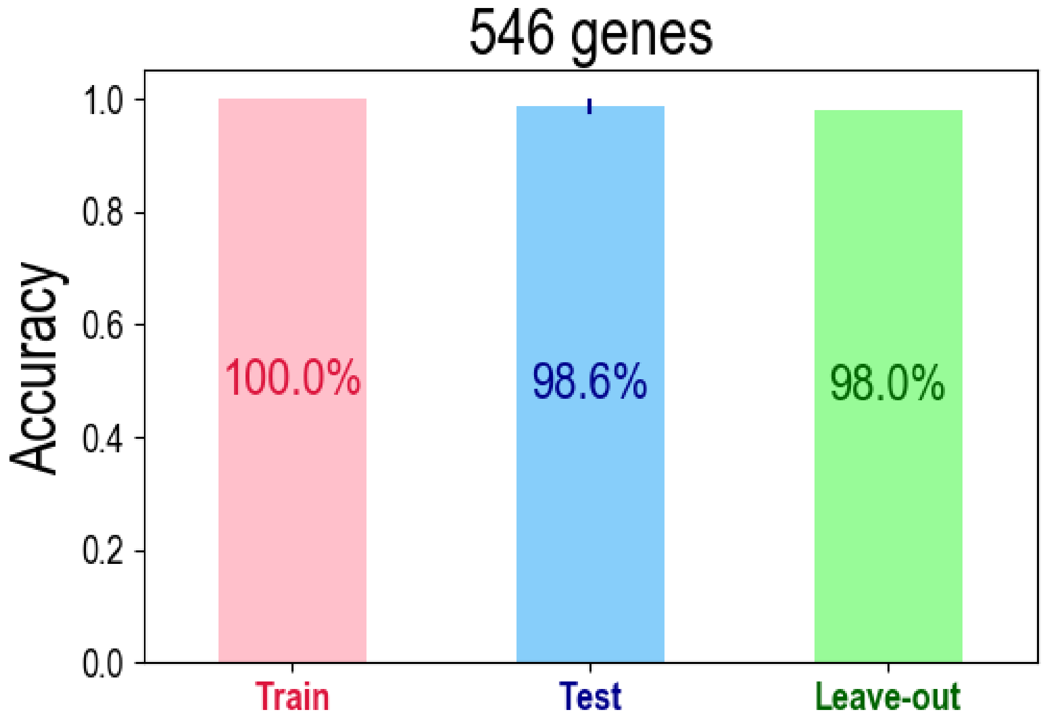


Training Set: 1640

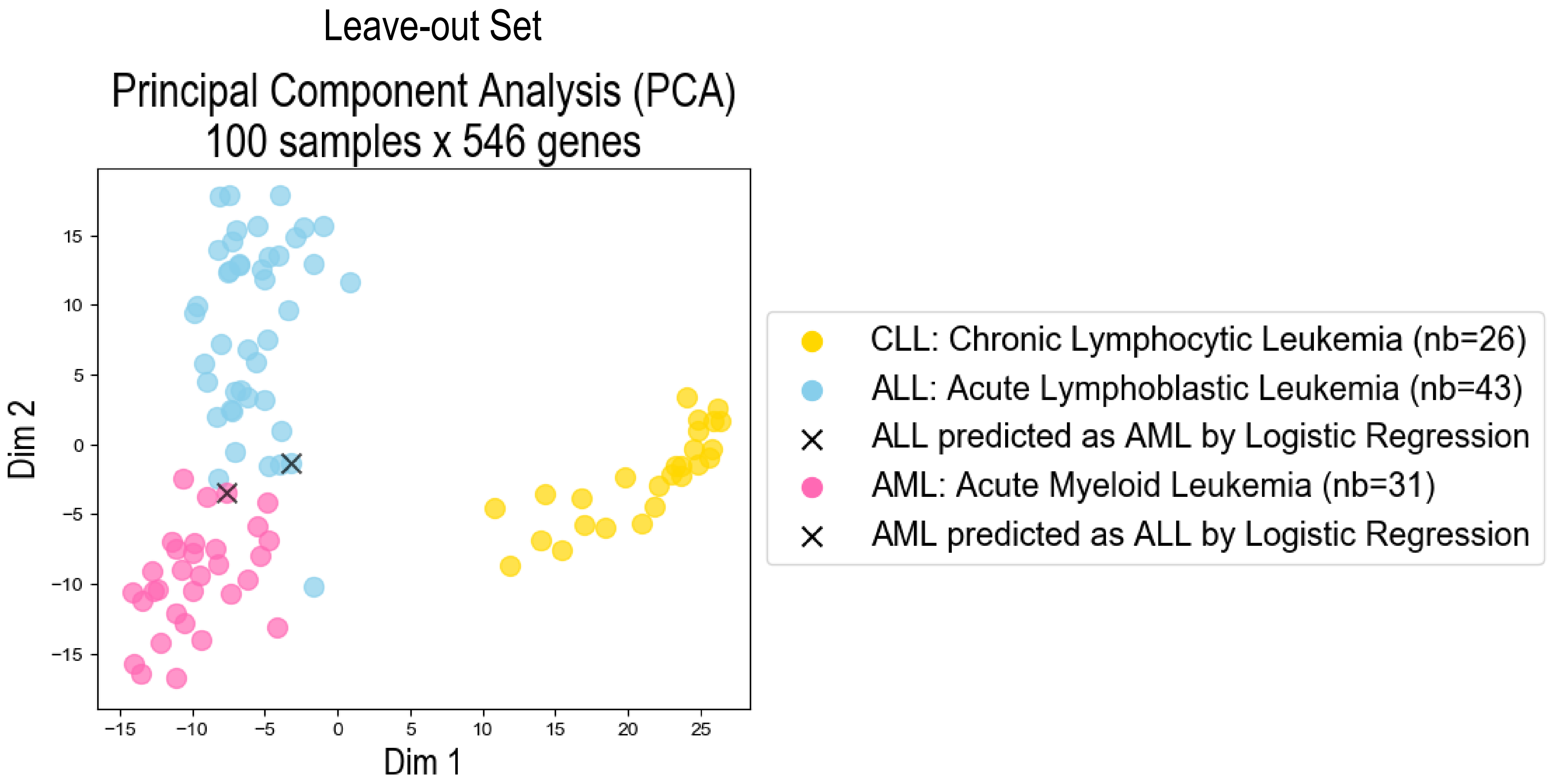
Leave-out Set: 100



Train	Train	Test
Test	Train	Train
Train	Test	Train

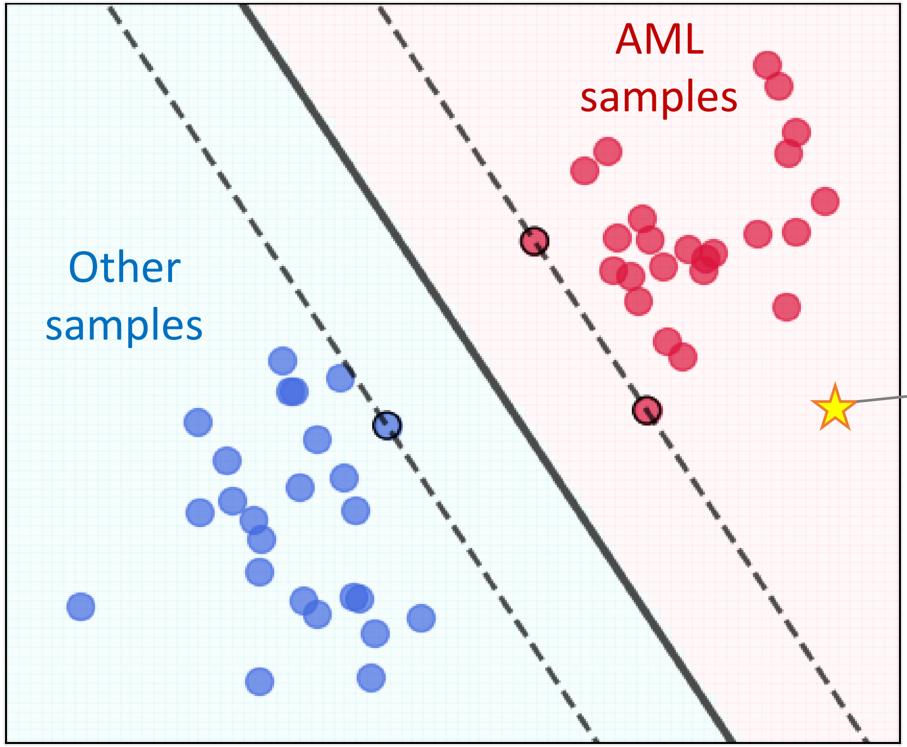


Train a classifier: Logistic Regression



Train a classifier: Support-vector machine (SVM)

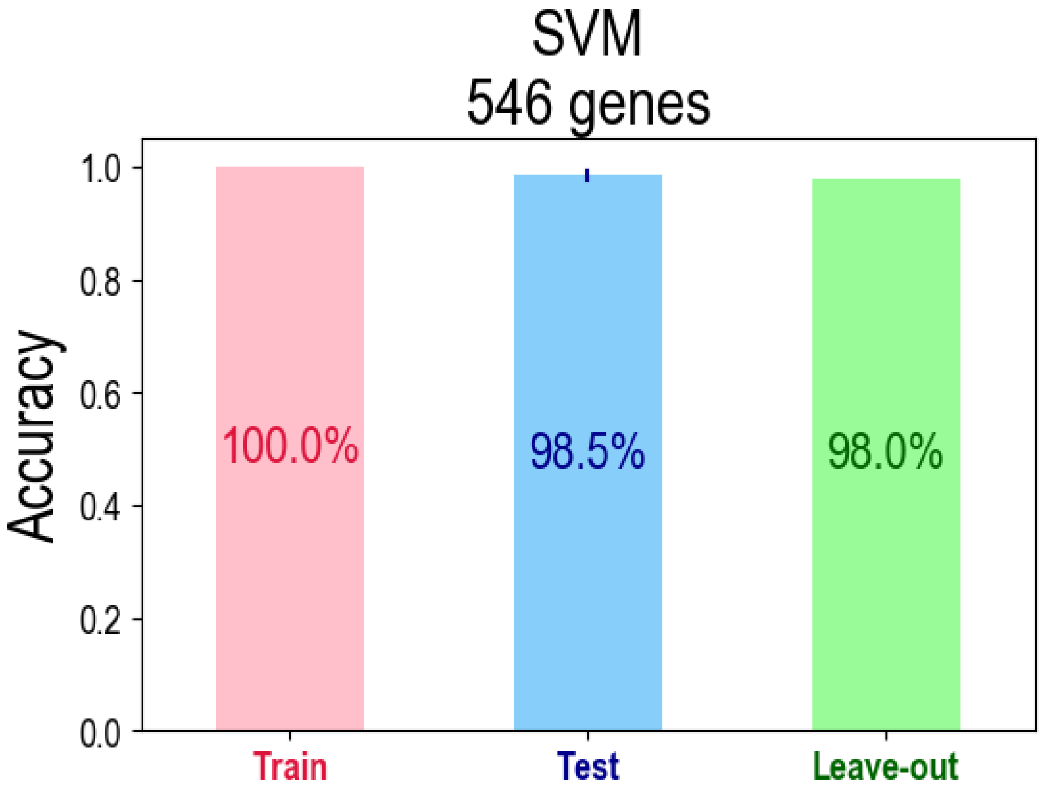
SVM method separates samples by a clear gap with the maximal possible margin.



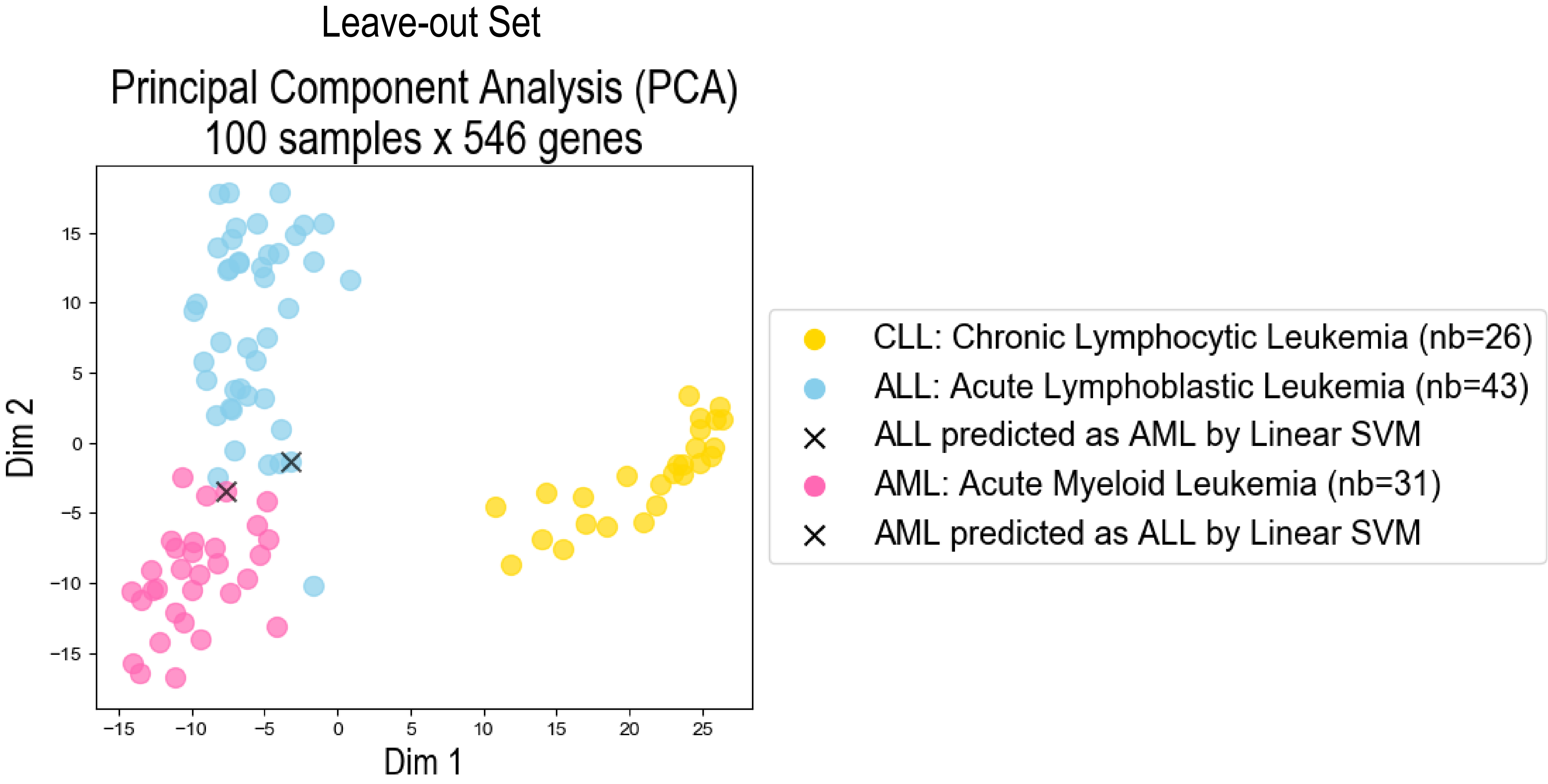
New sample:
will be classified as AML

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using a kernel, implicitly mapping their inputs into high-dimensional feature spaces.

Train a classifier: Linear SVM

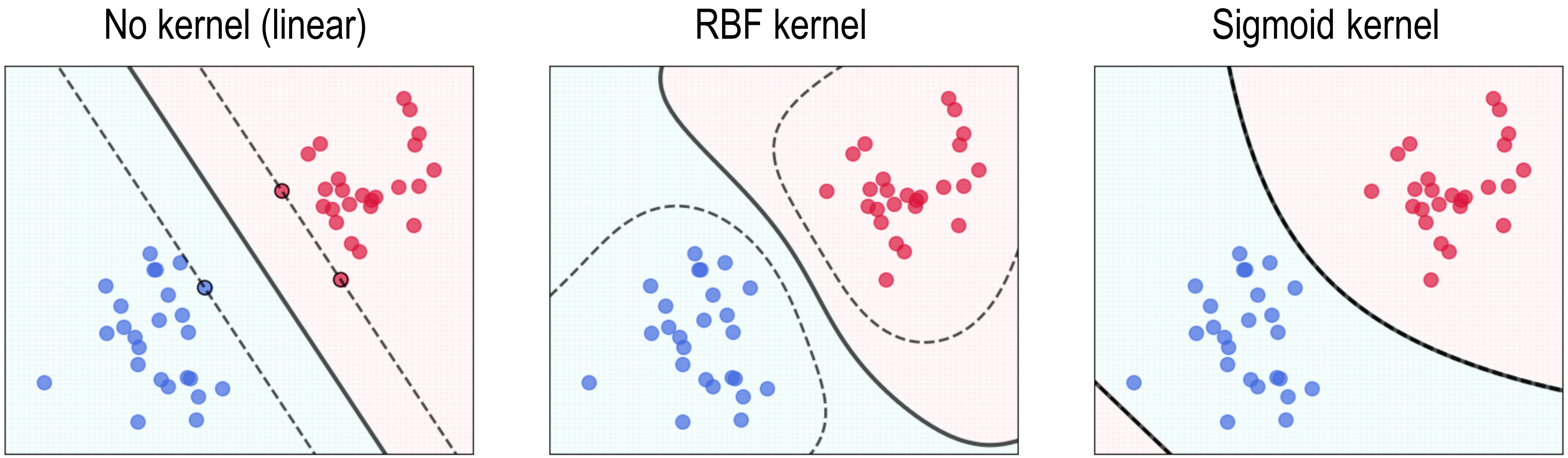


Train a classifier: Linear SVM



Train a classifier: Support-vector machine (SVM)

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using a kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

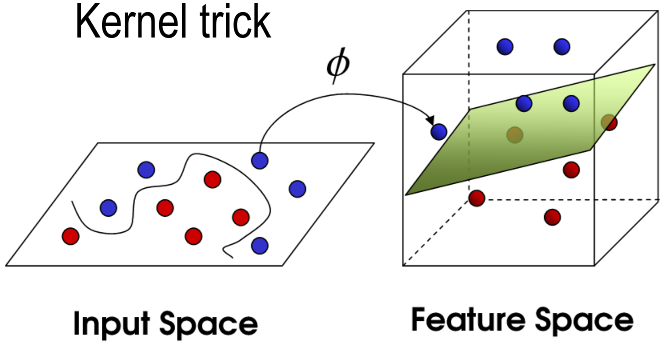


No kernel (linear)

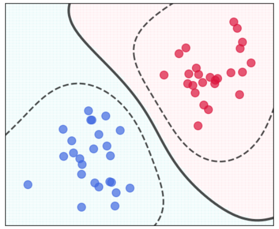
RBF kernel

Sigmoid kernel

Kernel trick



Train a classifier: Non-linear SVM with RBF kernel



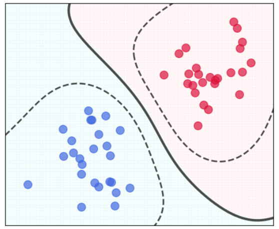
To train a non-linear SVM classifier we need to fix two parameters:

- Parameter C: degree of ridge regularization
  how smooth should our model be to avoid overfitting?

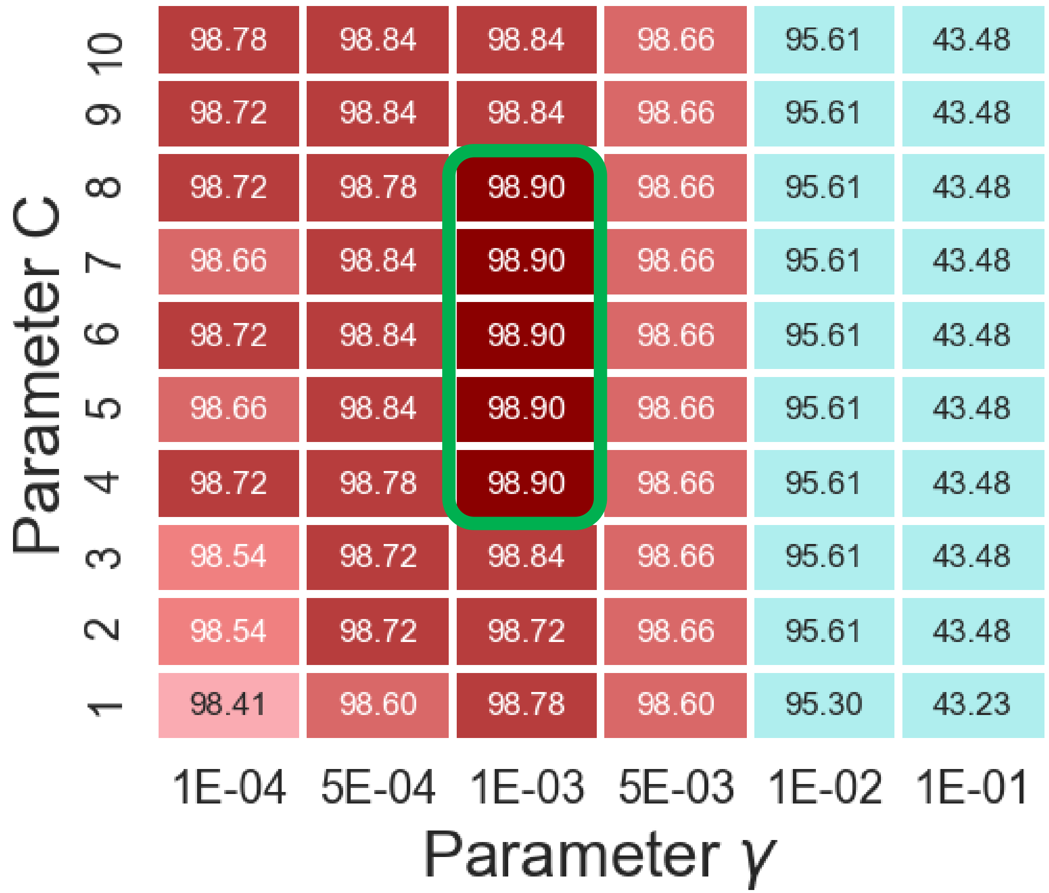
- Parameter γ: degree of non-linearity
  strongly non-linear model or almost linear?

```
classifier = SVC(kernel='rbf', C=?, gamma=?)
```

Train a classifier: Non-linear SVM with RBF kernel



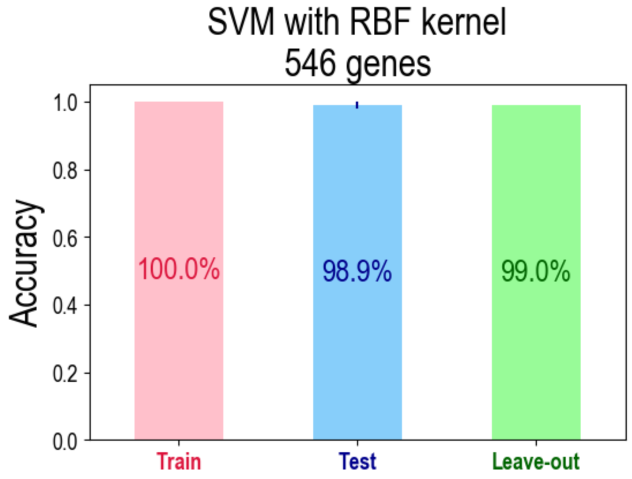
Grid search for optimal parameters using cross-validation



Parameter C	1E-04	5E-04	1E-03	5E-03	1E-02	1E-01
10	98.78	98.84	98.84	98.66	95.61	43.48
9	98.72	98.84	98.84	98.66	95.61	43.48
8	98.72	98.78	98.90	98.66	95.61	43.48
7	98.66	98.84	98.90	98.66	95.61	43.48
6	98.72	98.84	98.90	98.66	95.61	43.48
5	98.66	98.84	98.90	98.66	95.61	43.48
4	98.72	98.78	98.90	98.66	95.61	43.48
3	98.54	98.72	98.84	98.66	95.61	43.48
2	98.54	98.72	98.72	98.66	95.61	43.48
1	98.41	98.60	98.78	98.60	95.30	43.23

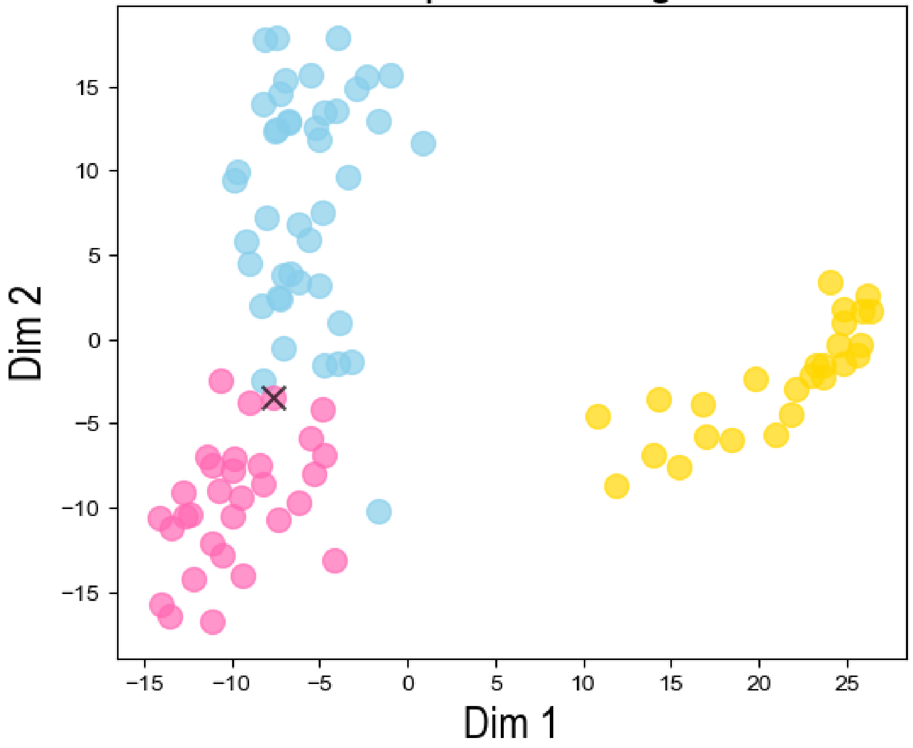
classifier = SVC(kernel='rbf', C=**6**, gamma=**1e-3**)

Train a classifier: Non-linear SVM with RBF kernel


SVM with RBF kernel
546 genes

Leave-out Set

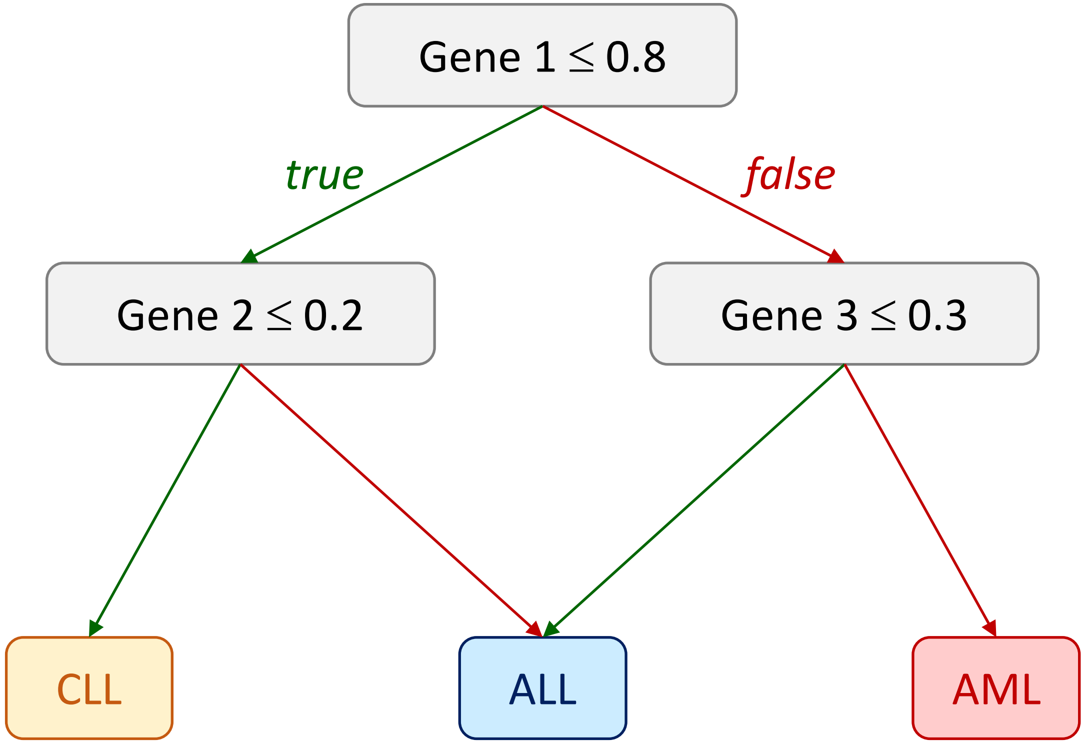
Principal Component Analysis (PCA)
100 samples x 546 genes



- ● CLL: Chronic Lymphocytic Leukemia (nb=26)
- ● ALL: Acute Lymphoblastic Leukemia (nb=43)
- ● AML: Acute Myeloid Leukemia (nb=31)
- × AML predicted as ALL by SVM with RBF kernel

Train a classifier: Random Forest

Random Forest creates several decision trees using random subsets of data. The final result is obtained by a vote from all decision trees.

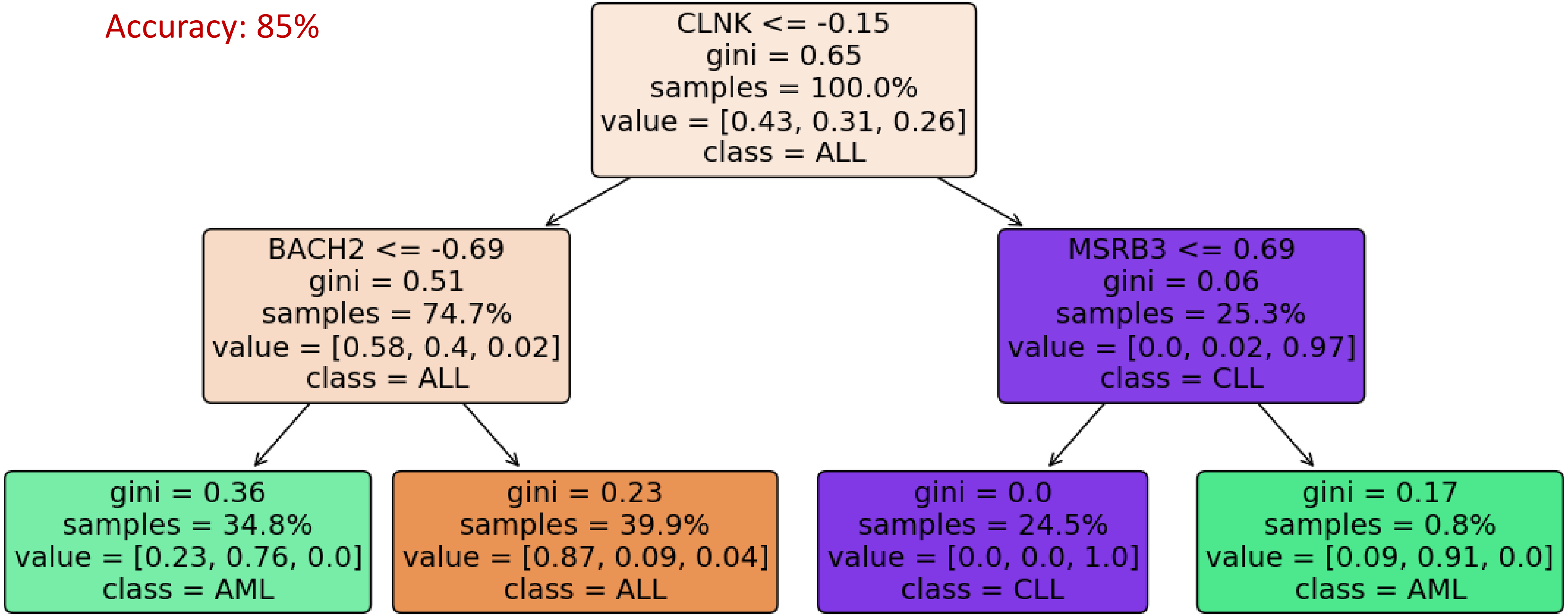


A simple decision tree

Train a classifier: Decision Tree

A simple forest composed of a unique tree:

Accuracy: 85%



CLNK <= -0.15
gini = 0.65
samples = 100.0%
value = [0.43, 0.31, 0.26]
class = ALL

BACH2 <= -0.69
gini = 0.51
samples = 74.7%
value = [0.58, 0.4, 0.02]
class = ALL

MSRB3 <= 0.69
gini = 0.06
samples = 25.3%
value = [0.0, 0.02, 0.97]
class = CLL

gini = 0.36
samples = 34.8%
value = [0.23, 0.76, 0.0]
class = AML

gini = 0.23
samples = 39.9%
value = [0.87, 0.09, 0.04]
class = ALL

gini = 0.0
samples = 24.5%
value = [0.0, 0.0, 1.0]
class = CLL

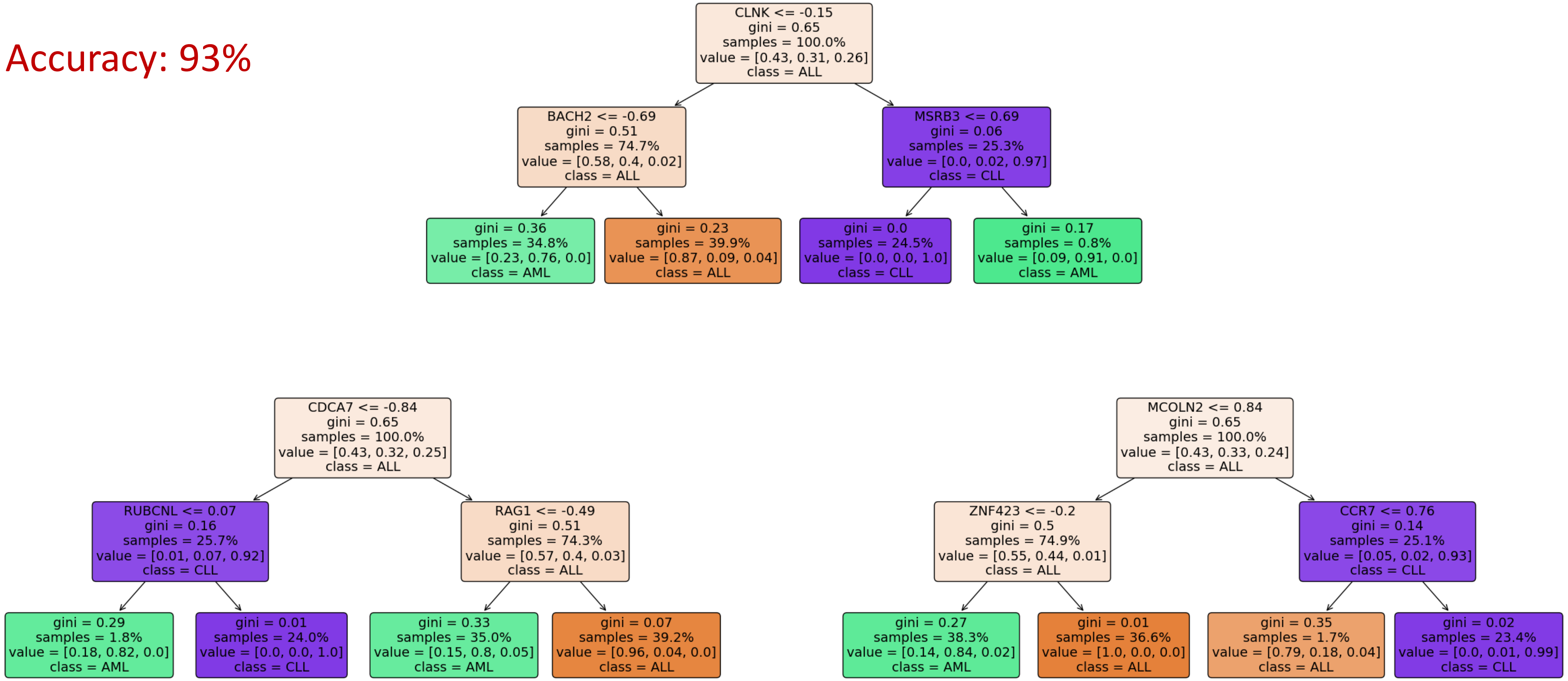
gini = 0.17
samples = 0.8%
value = [0.09, 0.91, 0.0]
class = AML

```
classifier = RandomForestClassifier(n_estimators=1, max_depth=2)
```

Train a classifier: Random Forest

A simple Random Forest:
number of trees = 3, depth of each tree = 2

Accuracy: 93%



CLNK <= -0.15
gini = 0.65
samples = 100.0%
value = [0.43, 0.31, 0.26]
class = ALL

BACH2 <= -0.69
gini = 0.51
samples = 74.7%
value = [0.58, 0.4, 0.02]
class = ALL

MSRB3 <= 0.69
gini = 0.06
samples = 25.3%
value = [0.0, 0.02, 0.97]
class = CLL

gini = 0.36
samples = 34.8%
value = [0.23, 0.76, 0.0]
class = AML

gini = 0.23
samples = 39.9%
value = [0.87, 0.09, 0.04]
class = ALL

gini = 0.0
samples = 24.5%
value = [0.0, 0.0, 1.0]
class = CLL

gini = 0.17
samples = 0.8%
value = [0.09, 0.91, 0.0]
class = AML

CDCA7 <= -0.84
gini = 0.65
samples = 100.0%
value = [0.43, 0.32, 0.25]
class = ALL

RUBCNL <= 0.07
gini = 0.16
samples = 25.7%
value = [0.01, 0.07, 0.92]
class = CLL

RAG1 <= -0.49
gini = 0.51
samples = 74.3%
value = [0.57, 0.4, 0.03]
class = ALL

gini = 0.29
samples = 1.8%
value = [0.18, 0.82, 0.0]
class = AML

gini = 0.01
samples = 24.0%
value = [0.0, 0.0, 1.0]
class = CLL

gini = 0.33
samples = 35.0%
value = [0.15, 0.8, 0.05]
class = AML

gini = 0.07
samples = 39.2%
value = [0.96, 0.04, 0.0]
class = ALL

MCOLN2 <= 0.84
gini = 0.65
samples = 100.0%
value = [0.43, 0.33, 0.24]
class = ALL

ZNF423 <= -0.2
gini = 0.5
samples = 74.9%
value = [0.55, 0.44, 0.01]
class = ALL

CCR7 <= 0.76
gini = 0.14
samples = 25.1%
value = [0.05, 0.02, 0.93]
class = CLL

gini = 0.27
samples = 38.3%
value = [0.14, 0.84, 0.02]
class = AML

gini = 0.01
samples = 36.6%
value = [1.0, 0.0, 0.0]
class = ALL

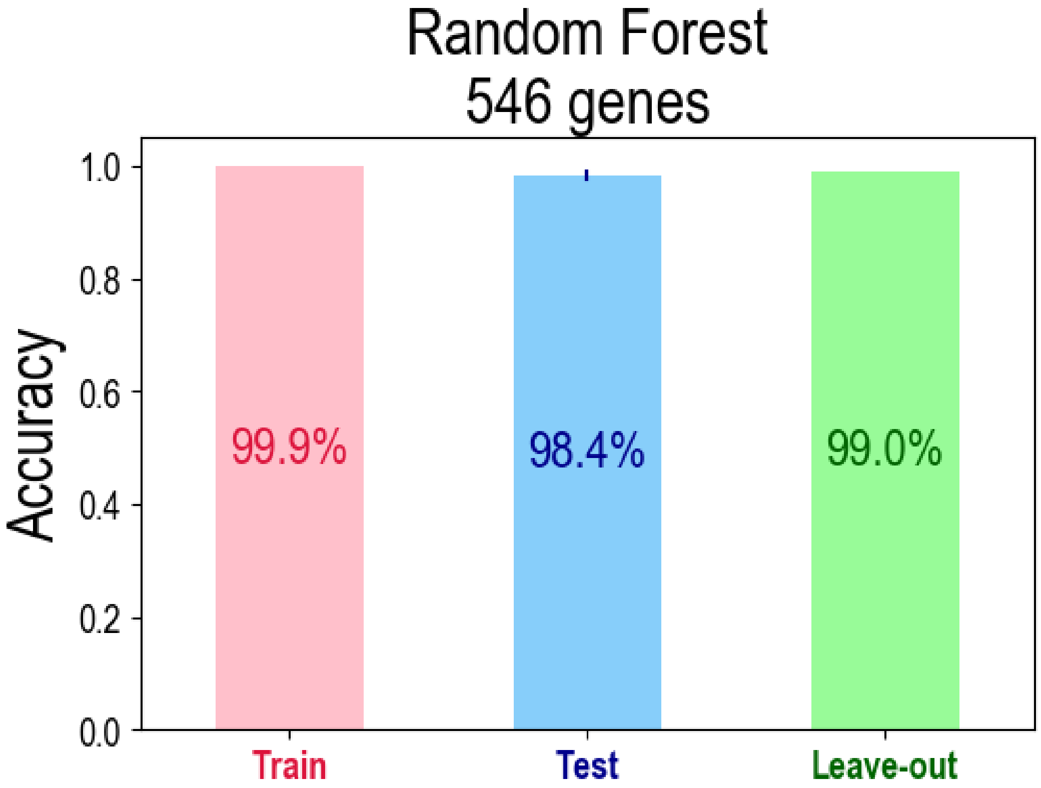
gini = 0.35
samples = 1.7%
value = [0.79, 0.18, 0.04]
class = ALL

gini = 0.02
samples = 23.4%
value = [0.0, 0.01, 0.99]
class = CLL

```
classifier = RandomForestClassifier(n_estimators=3, max_depth=2)
```

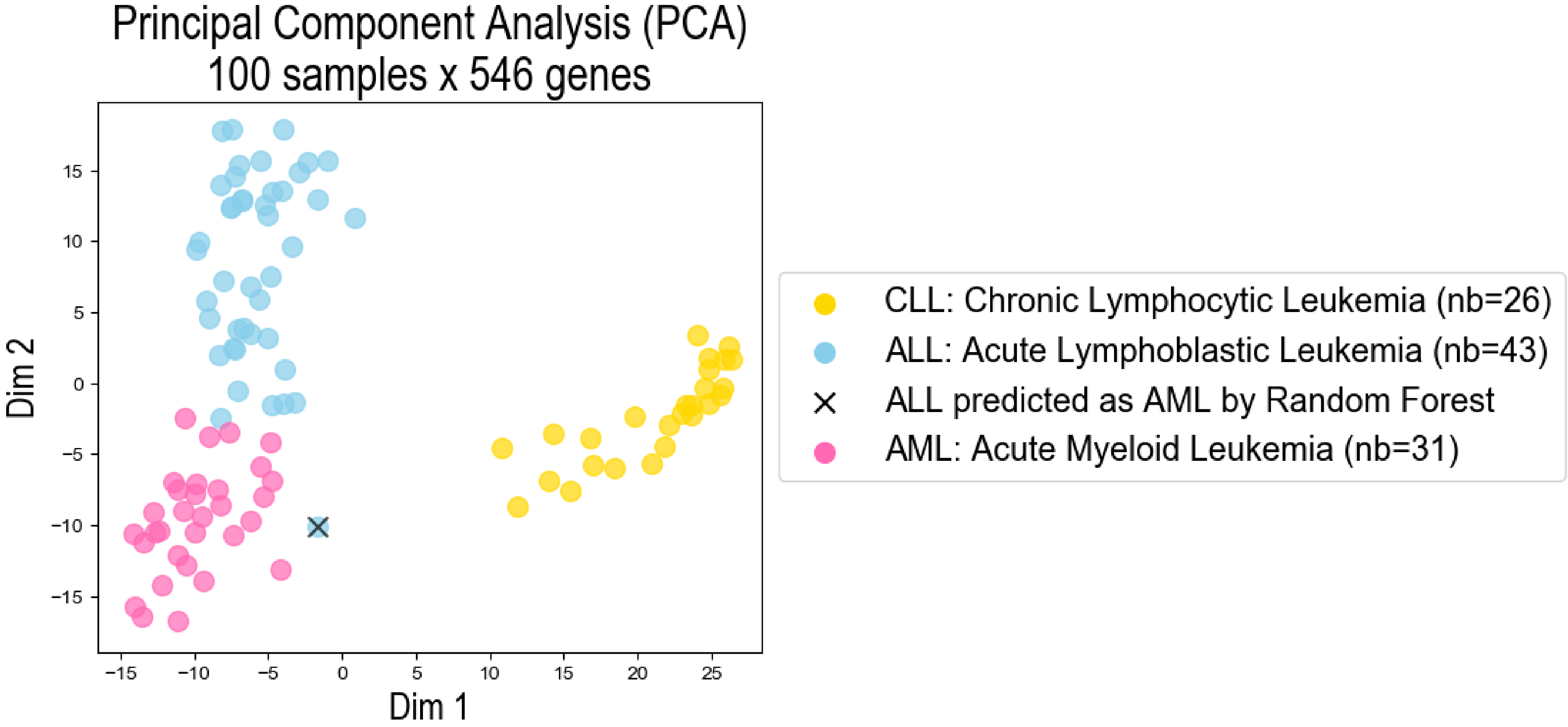
Train a classifier: Random Forest

Definitive Random Forest model:
number of trees = 50, depth of each tree = 8

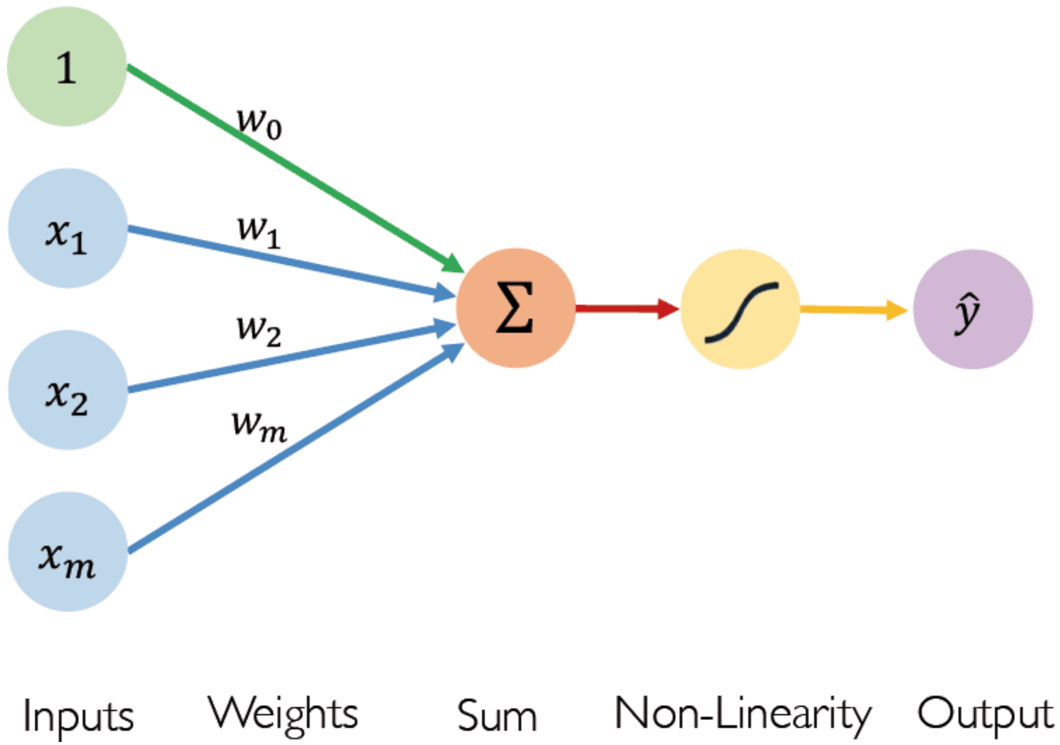


```
classifier = RandomForestClassifier(n_estimators=50, max_depth=8)
```

Train a classifier: Random Forest



Introduction to neural networks: The Perceptron



Output

Linear combination of inputs

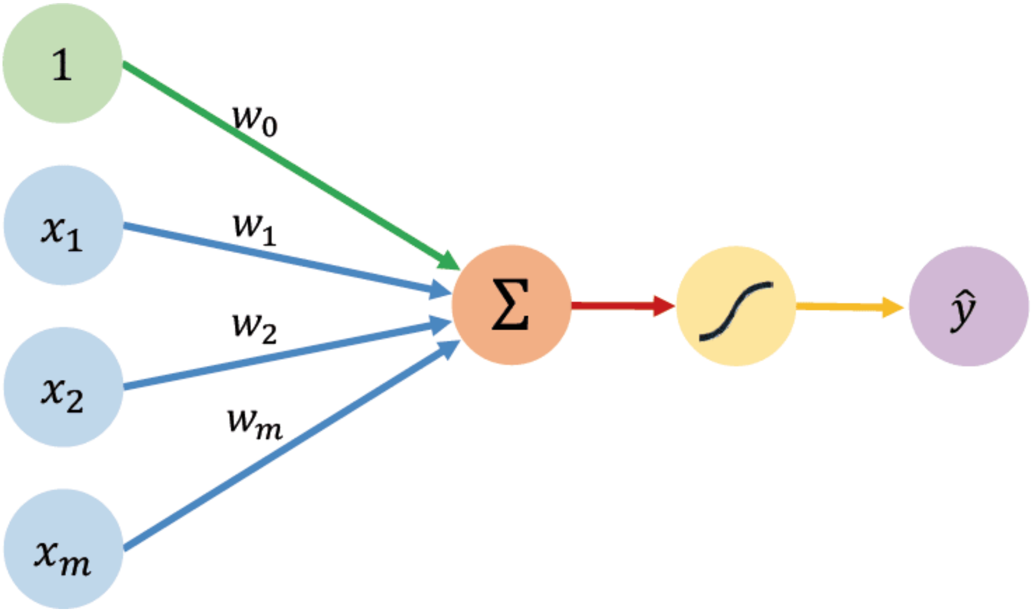
$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i \, w_i\right)$$

Non-linear activation function

Bias

Source: MIT Course 6.S191 Introduction to Deep Learning

Introduction to neural networks: The Perceptron



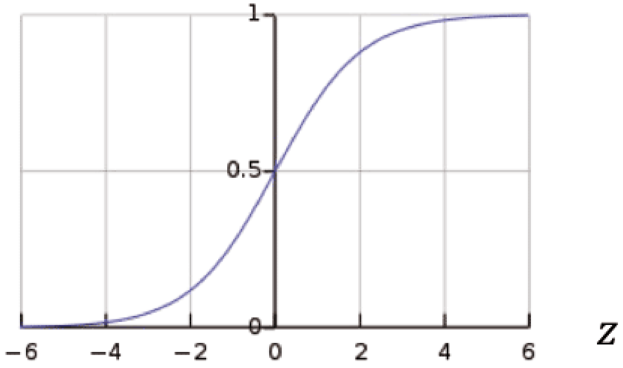
Train a perceptron with sigmoid activation function is mathematically equivalent to train a Logistic Regression.

Activation Functions

$$\hat{y} = g(\, w_0 + \boldsymbol{X}^T \boldsymbol{W} \,)$$

- Example: sigmoid function

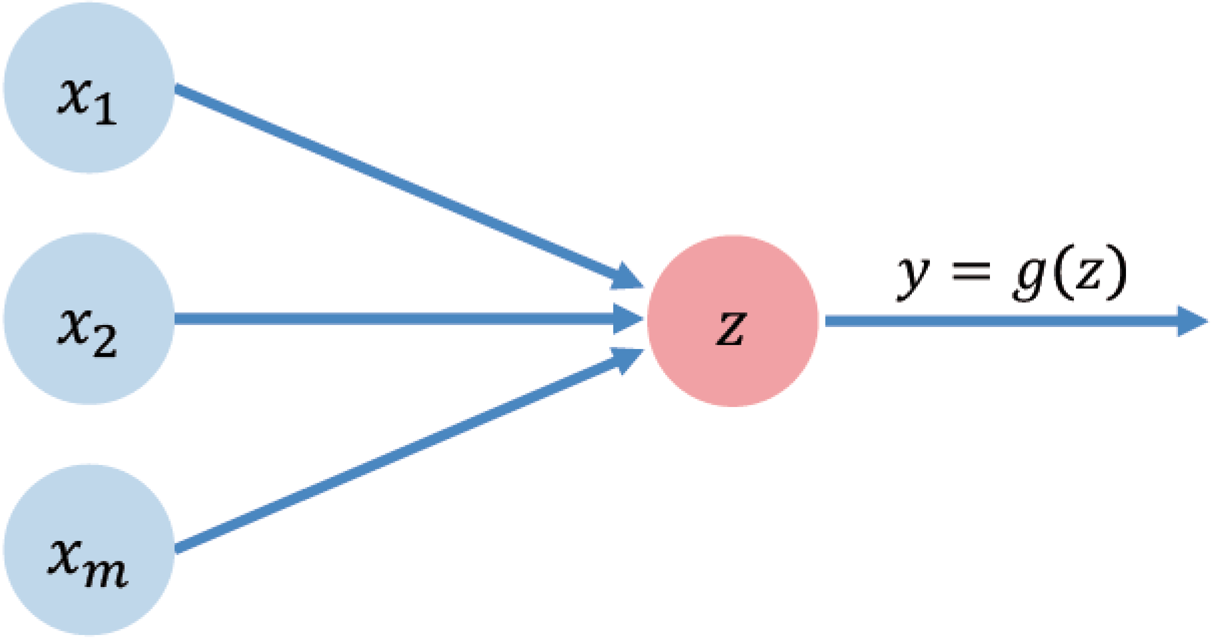
$$g(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$



Source: MIT Course 6.S191 Introduction to Deep Learning

Multi-layer perceptron (Neural Network)

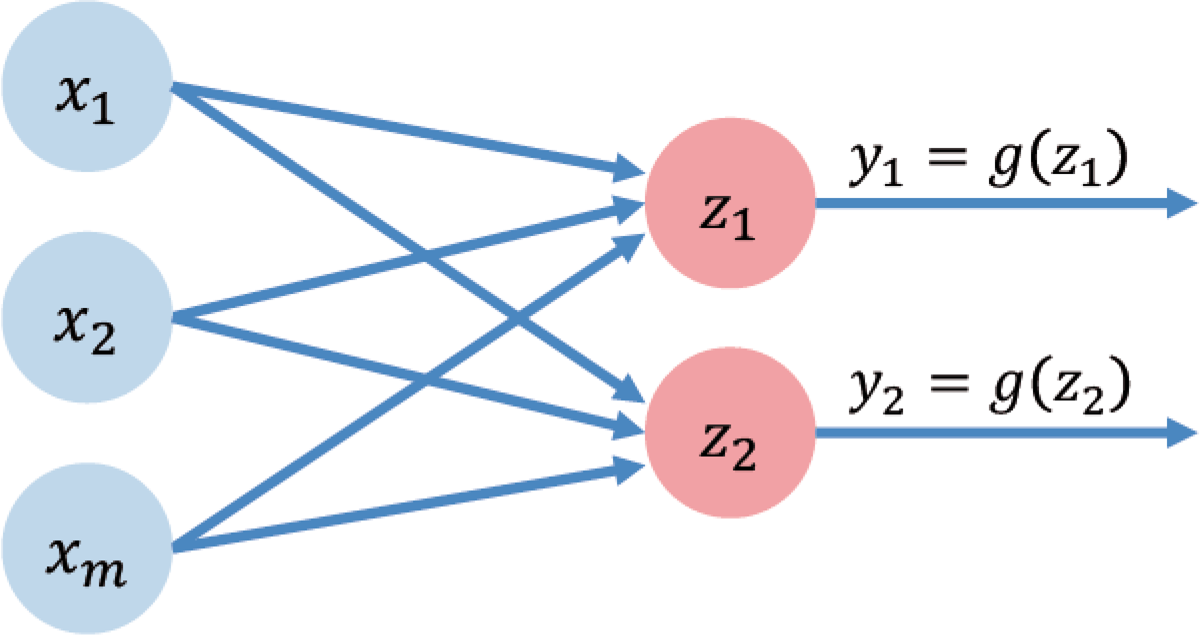
Simplified representation of a perceptron



Source: MIT Course 6.S191 Introduction to Deep Learning

Multi-layer perceptron (Neural Network)

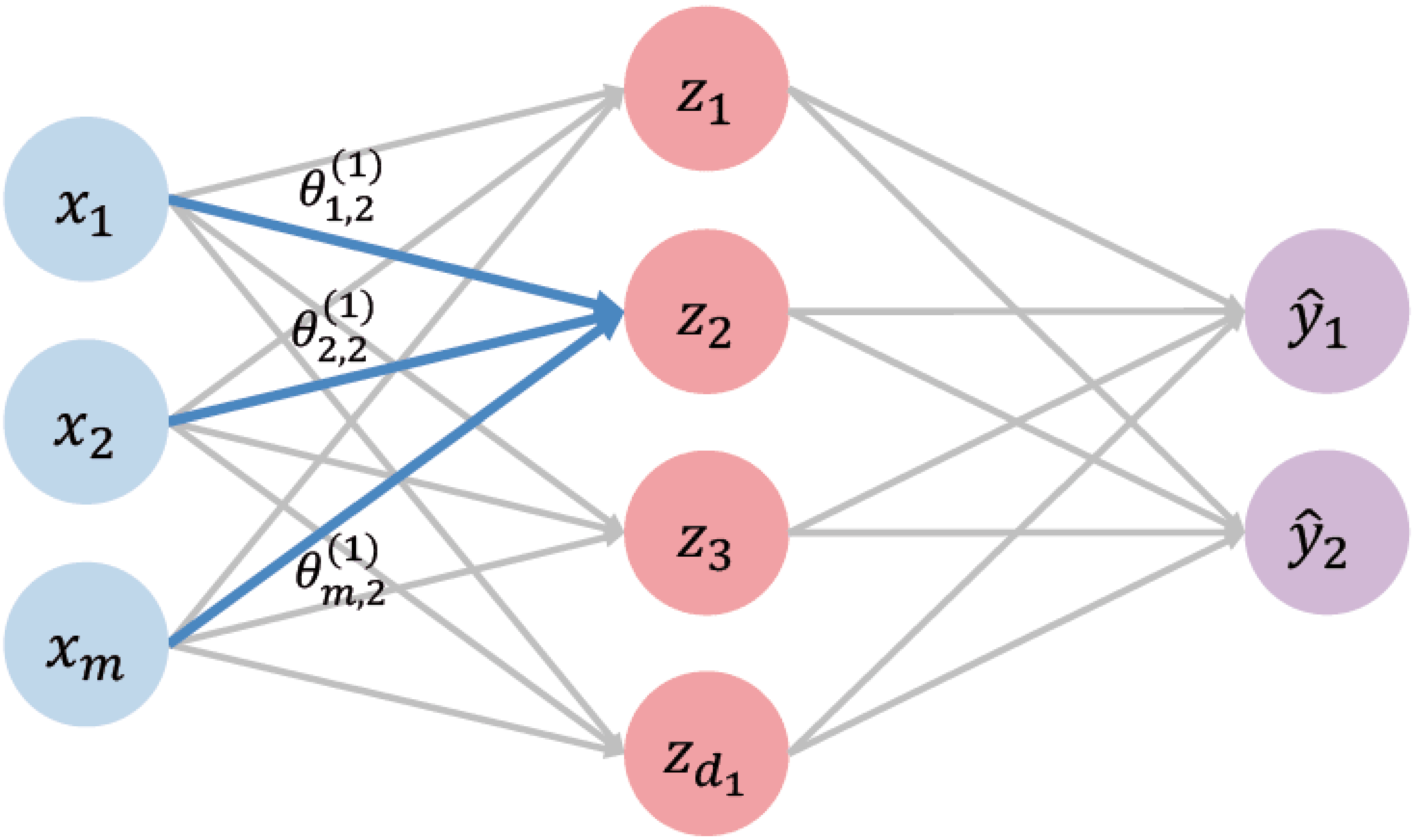
Perceptron with two outputs



Source: MIT Course 6.S191 Introduction to Deep Learning

Multi-layer perceptron (Neural Network)

Single layer neural network



Inputs

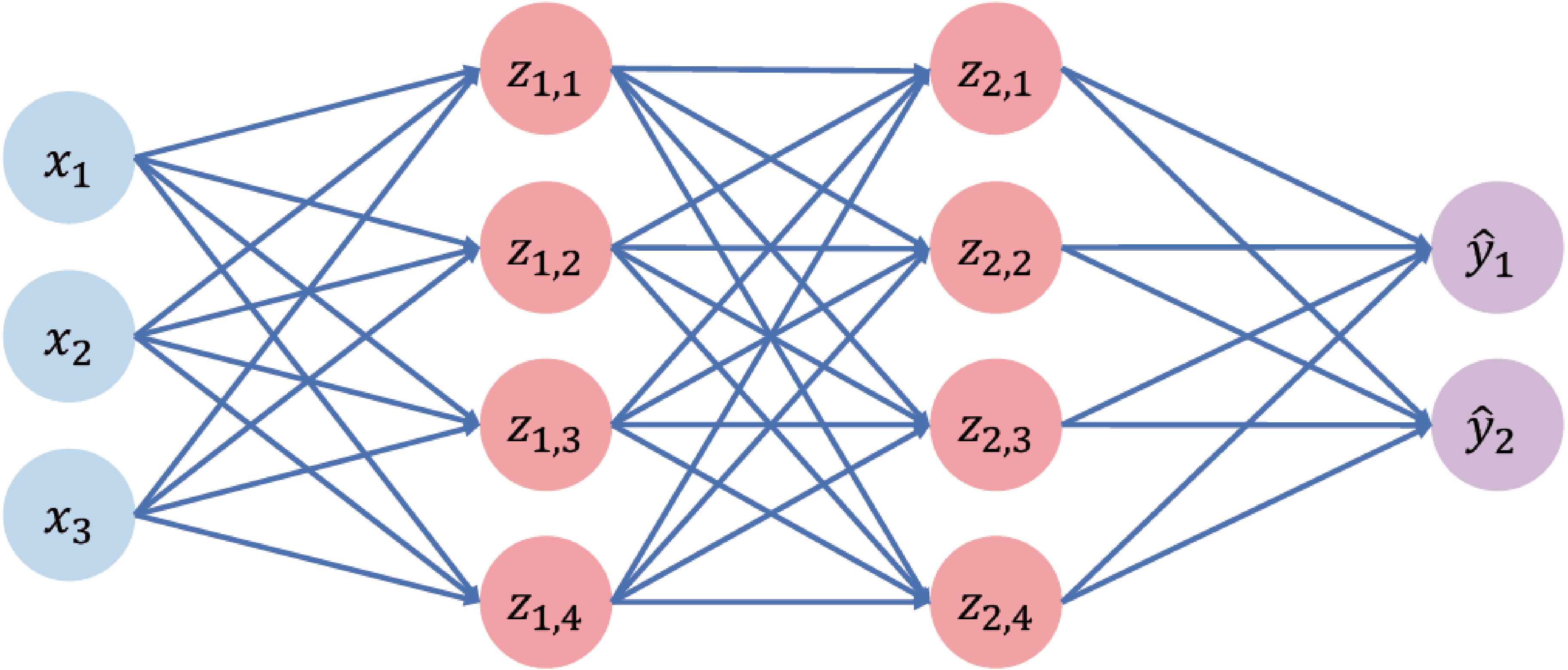
Hidden layer

Output

Source: MIT Course 6.S191 Introduction to Deep Learning

Multi-layer perceptron (Neural Network)

Deep neural network



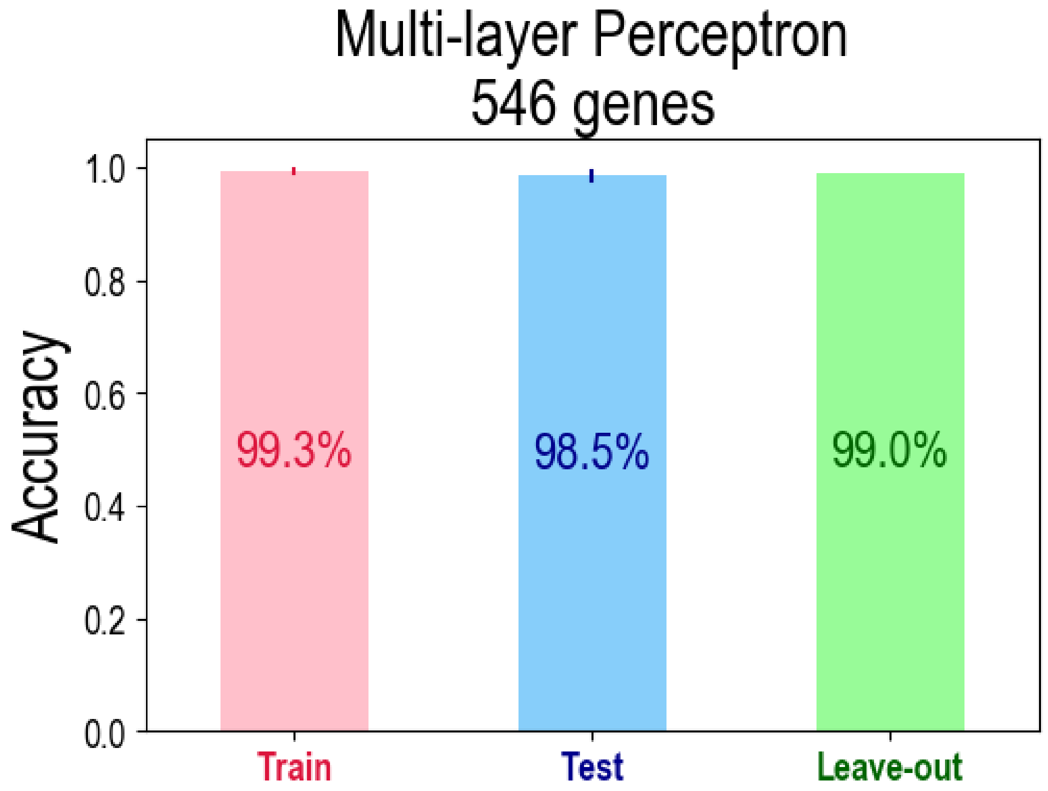
Inputs … Many hidden layers … Output

Source: MIT Course 6.S191 Introduction to Deep Learning

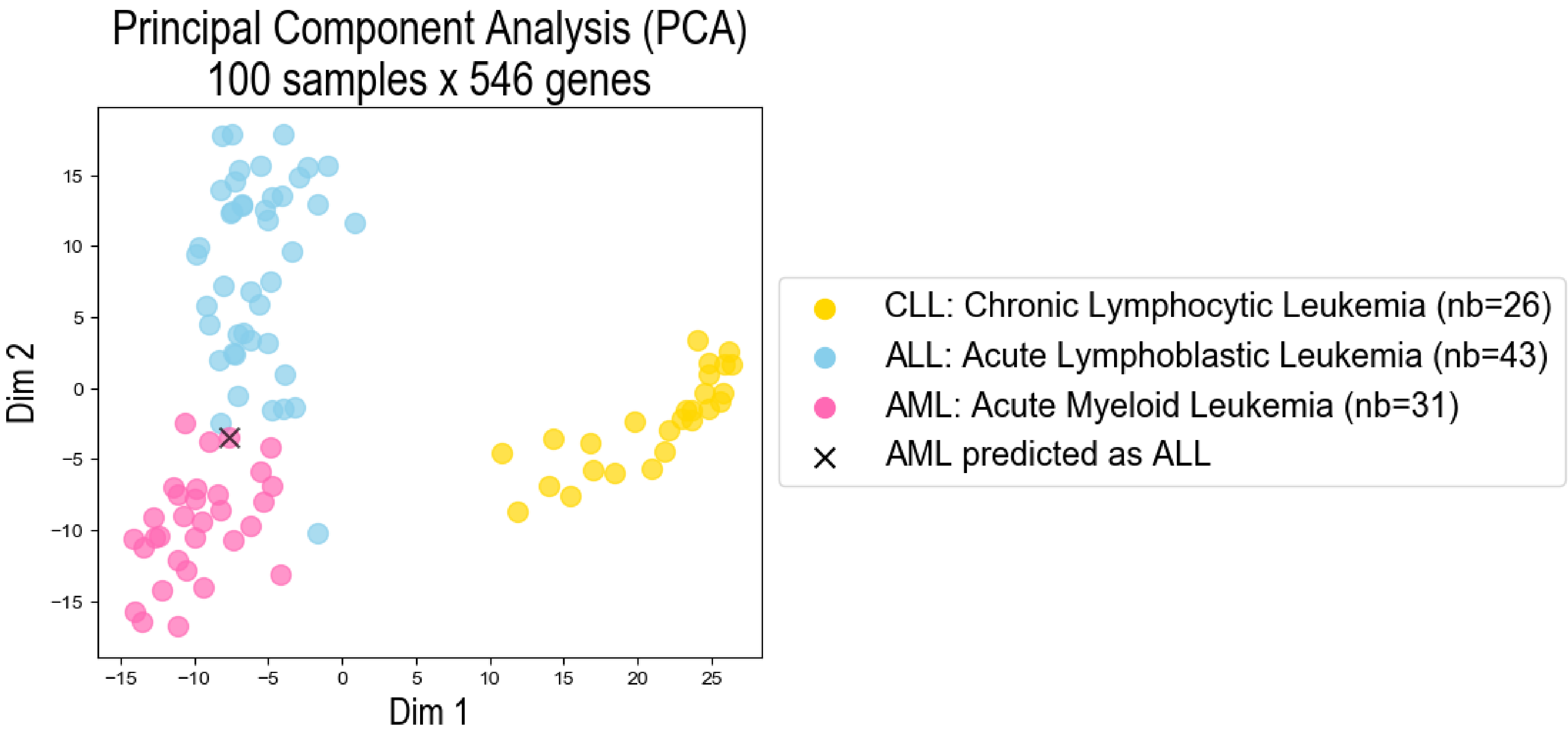
Multi-layer perceptron in Python

```python
from sklearn.neural_network import MLPClassifier
from sklearn import metrics

# Create a classifier
classifier = MLPClassifier(early_stopping=True, alpha=1.0,
                  hidden_layer_sizes=(100, 100, 100, 100))

# Train classifier using training dataset
classifier.fit(X_train_scaled, y_train)

# Predict labels for test dataset
y_pred_test = classifier.predict(X_test_scaled)

# Calculate accuracy comparing prediction
# with real labels in test dataset
accuracy = metrics.accuracy_score(y_test, y_pred_test)
```

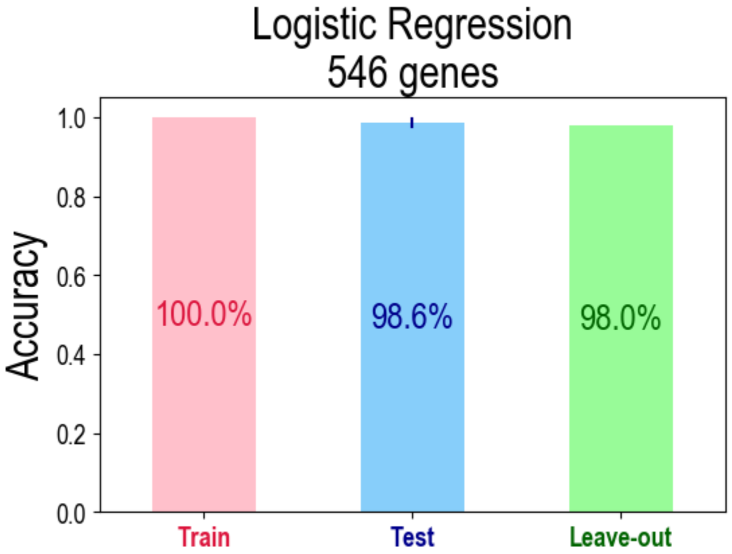
Train a classifier: Multi-layer Perceptron (Neural Network)

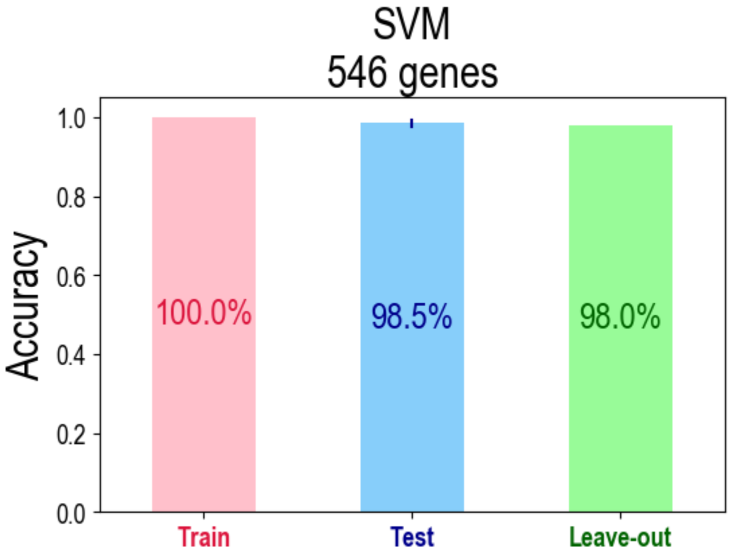


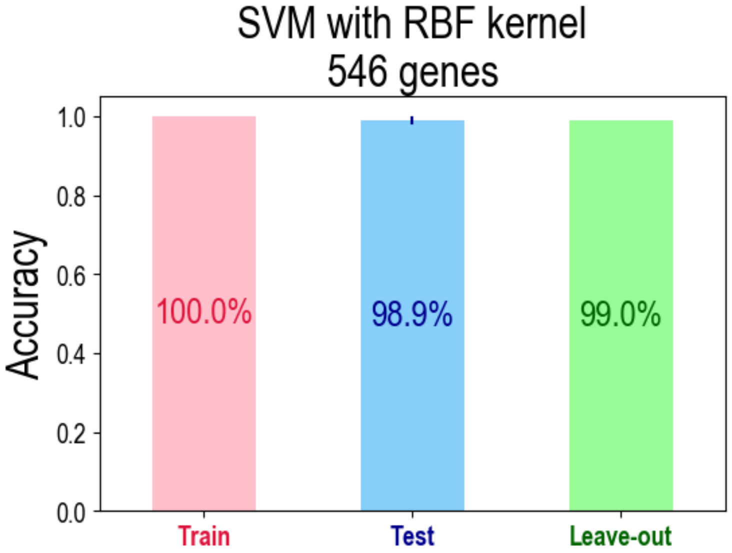
Train a classifier: Multi-layer Perceptron (Neural Network)

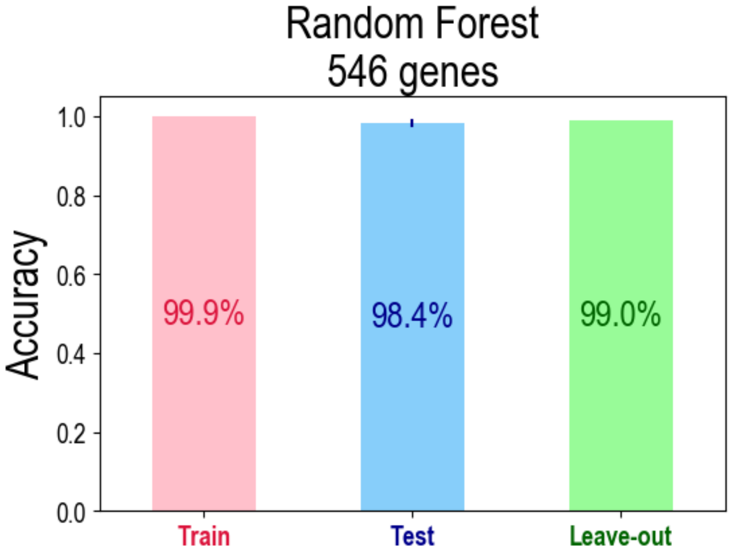


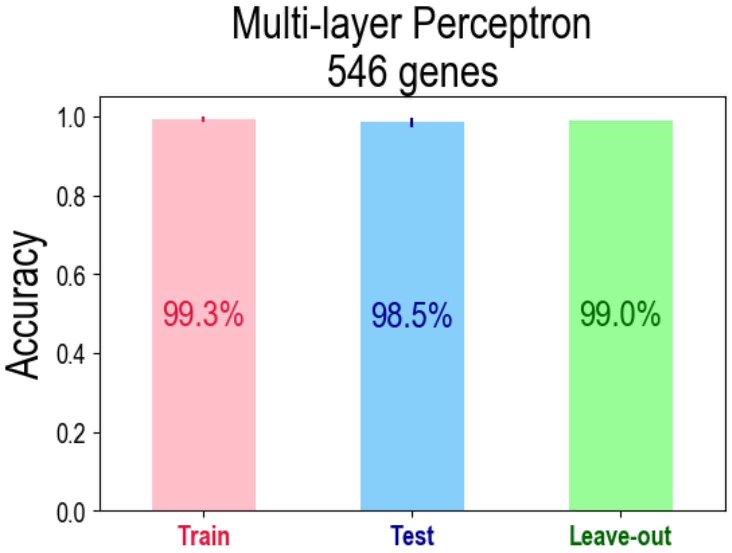
Benchmark


Logistic Regression
546 genes

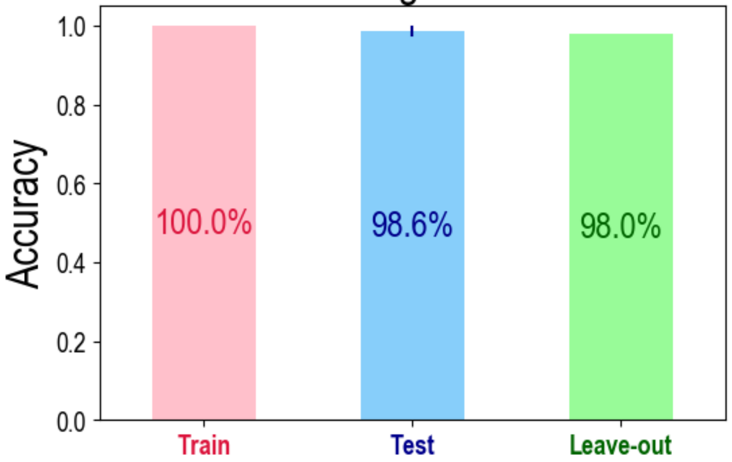

SVM
546 genes


SVM with RBF kernel
546 genes

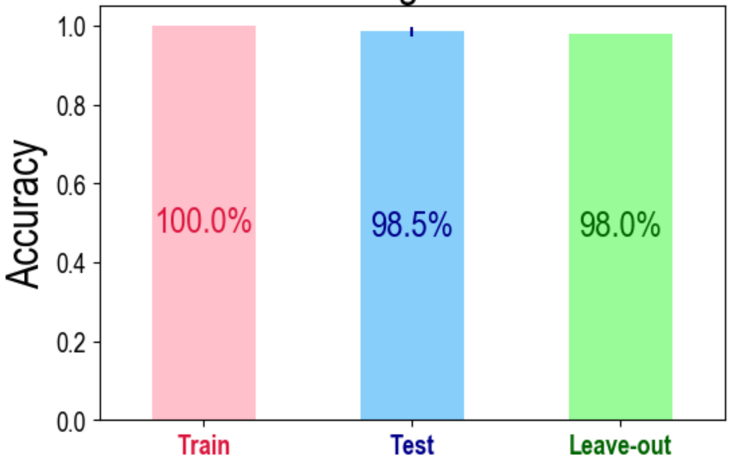

Random Forest
546 genes


Multi-layer Perceptron
546 genes

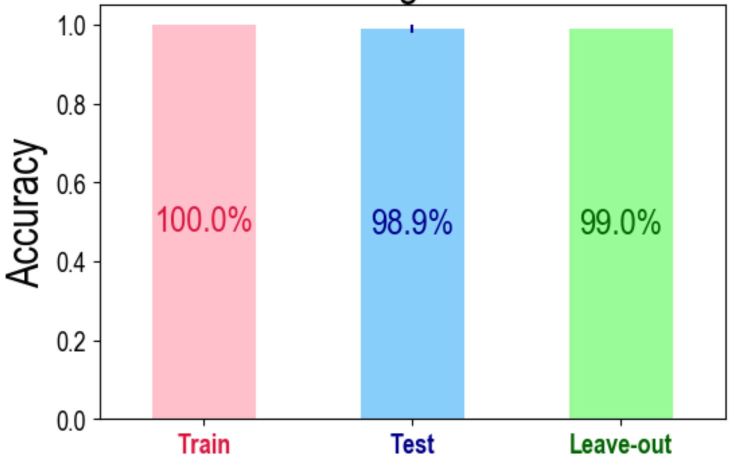
Benchmark



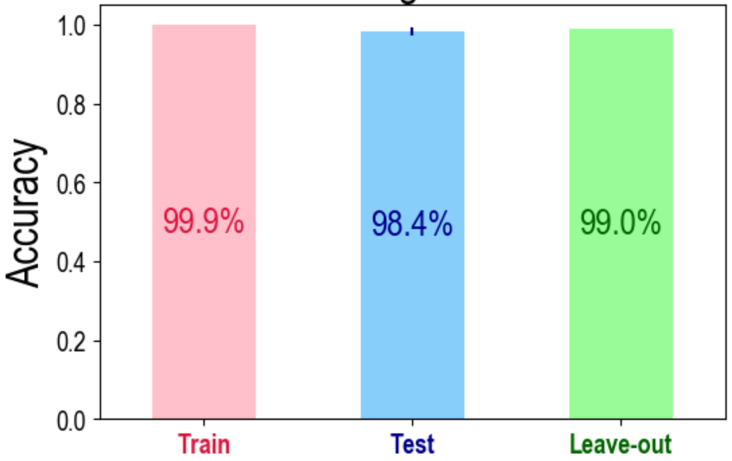
Logistic Regression
546 genes



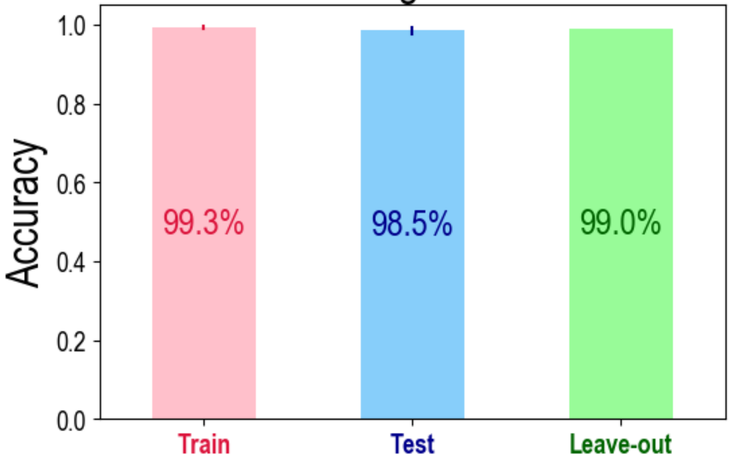
SVM
546 genes



SVM with RBF kernel
546 genes



Random Forest
546 genes



Multi-layer Perceptron
546 genes