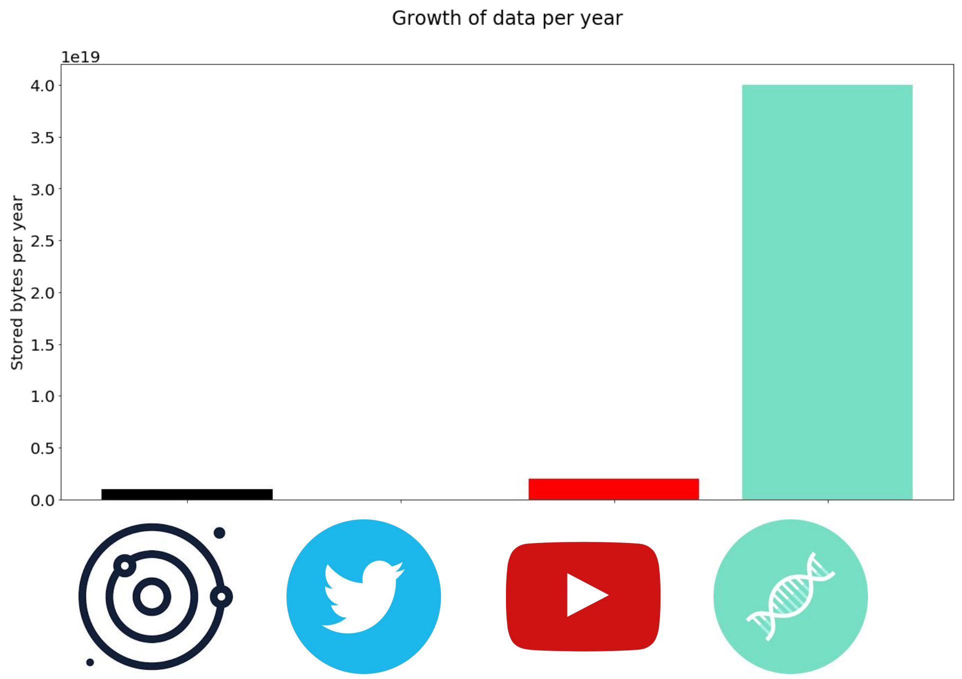
Introduction to Machine Learning for Bioinformatics

Sonika Tyagi, Navya Tyagi and Tyrone Chen
RMIT University Australia

12.11.2023

INCOB2023, @tyagilab

Main contributor to Big data growth is genomics



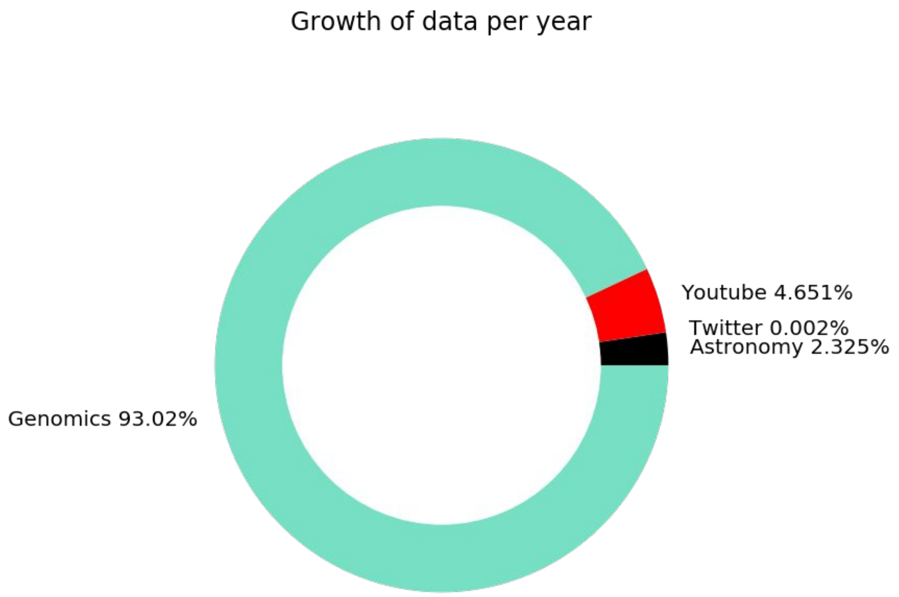


Fig by Tyrone Chen @tyagilab using data from Stephens et al, 2015

REGULATORY OMICS SIGNATURES DRIVE FUNCTIONAL OMICS SIGNATURES

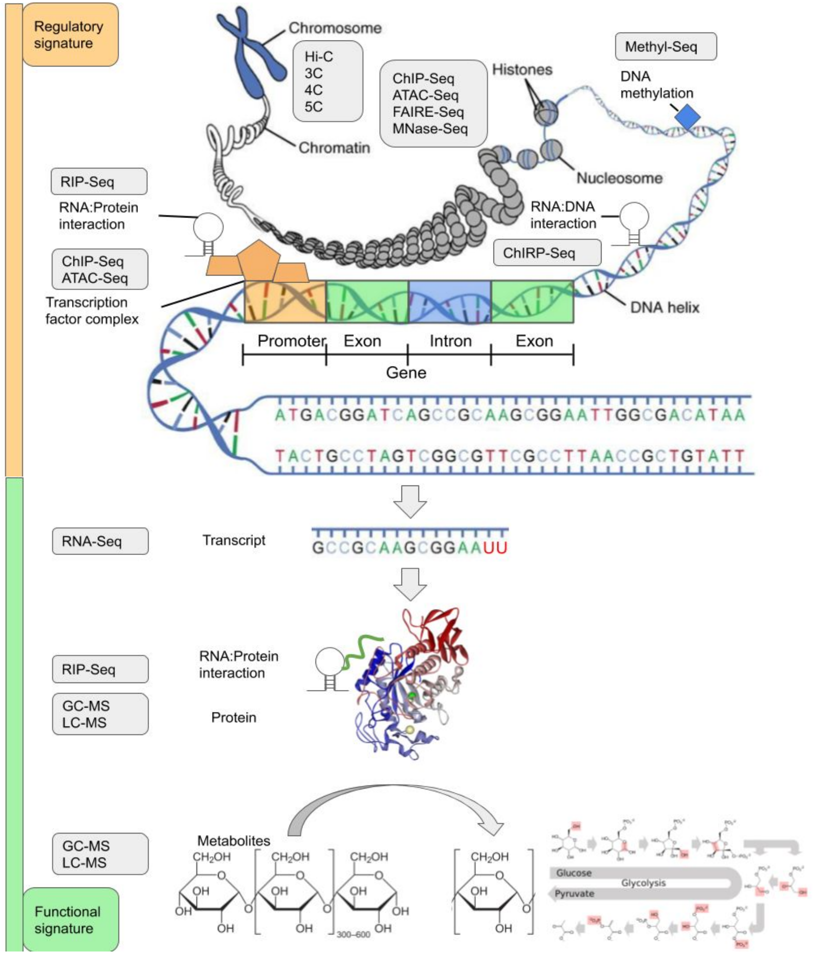
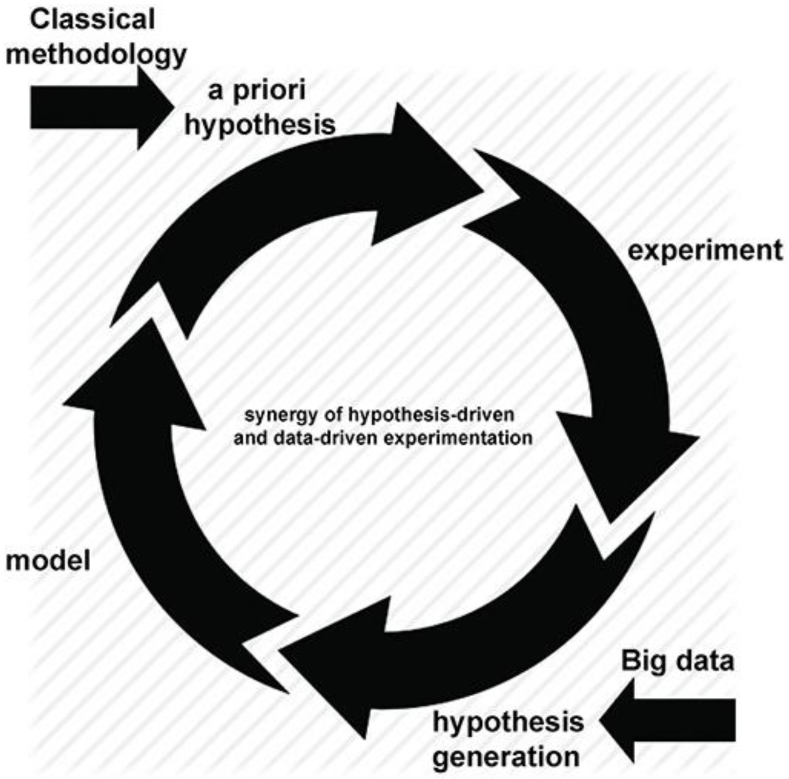


Image source: TyagiLab 2020

Genomics as a data-driven discipline



https://doi.org/10.3389/fmed.2019.00034

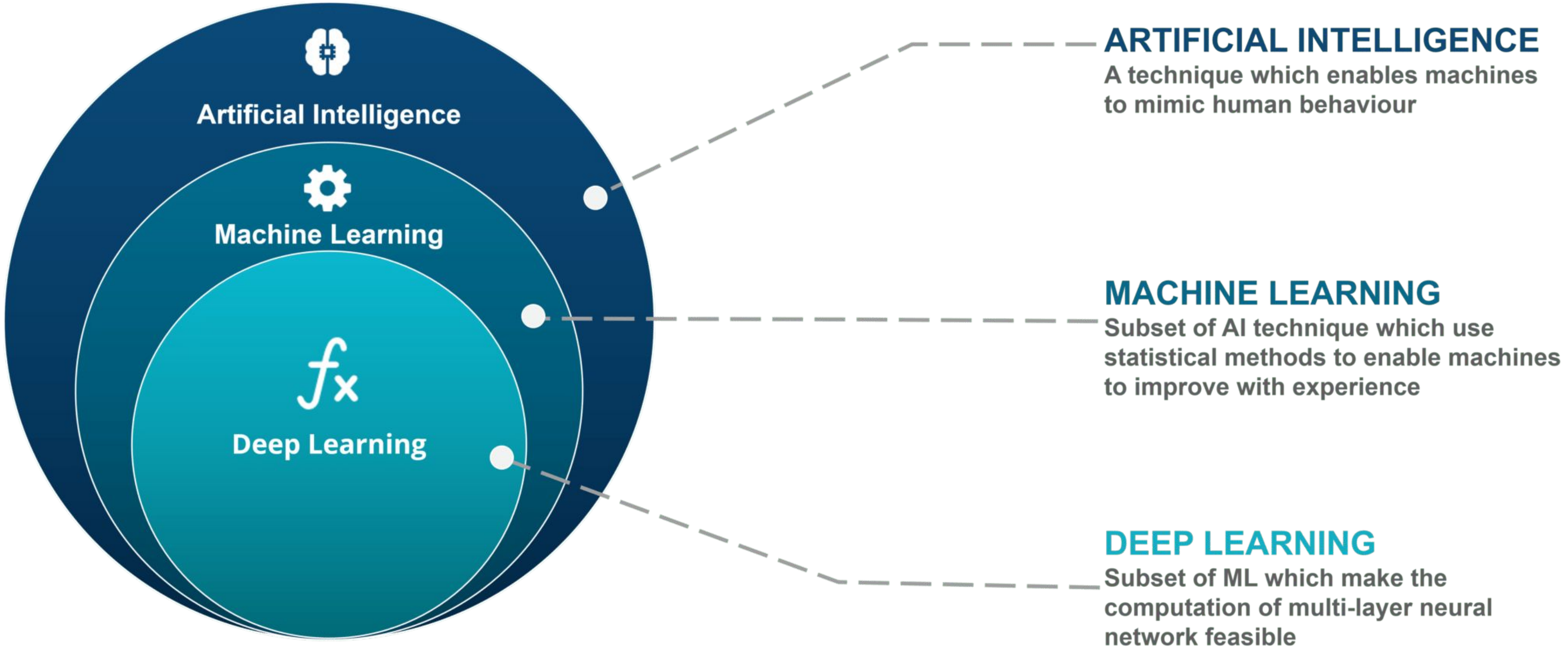
What is machine learning (ML) and Artificial intelligence (AI)?

AI = making intelligent machines

“Machines are intelligent to the extent that their actions can be expected to achieve their objectives”

-Prof Stuart Russell

Machine learning (ML) ?



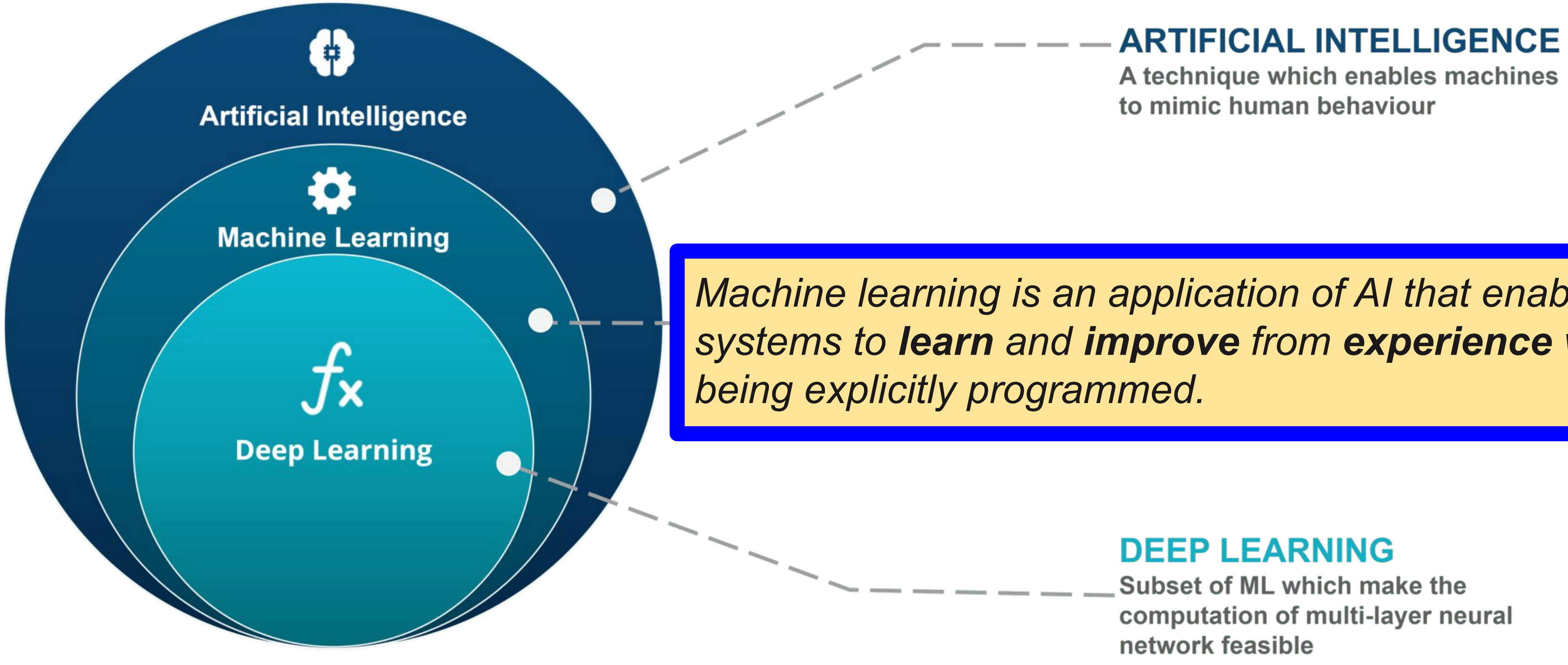
ARTIFICIAL INTELLIGENCE
A technique which enables machines to mimic human behaviour

MACHINE LEARNING
Subset of AI technique which use statistical methods to enable machines to improve with experience

DEEP LEARNING
Subset of ML which make the computation of multi-layer neural network feasible

Image source Eureka

Machine learning (ML) ?

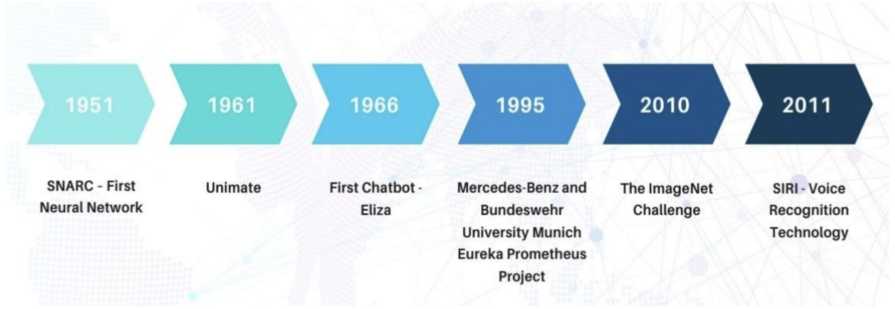


Artificial Intelligence
Machine Learning
Deep Learning
ARTIFICIAL INTELLIGENCE
A technique which enables machines to mimic human behaviour
DEEP LEARNING
Subset of ML which make the computation of multi-layer neural network feasible

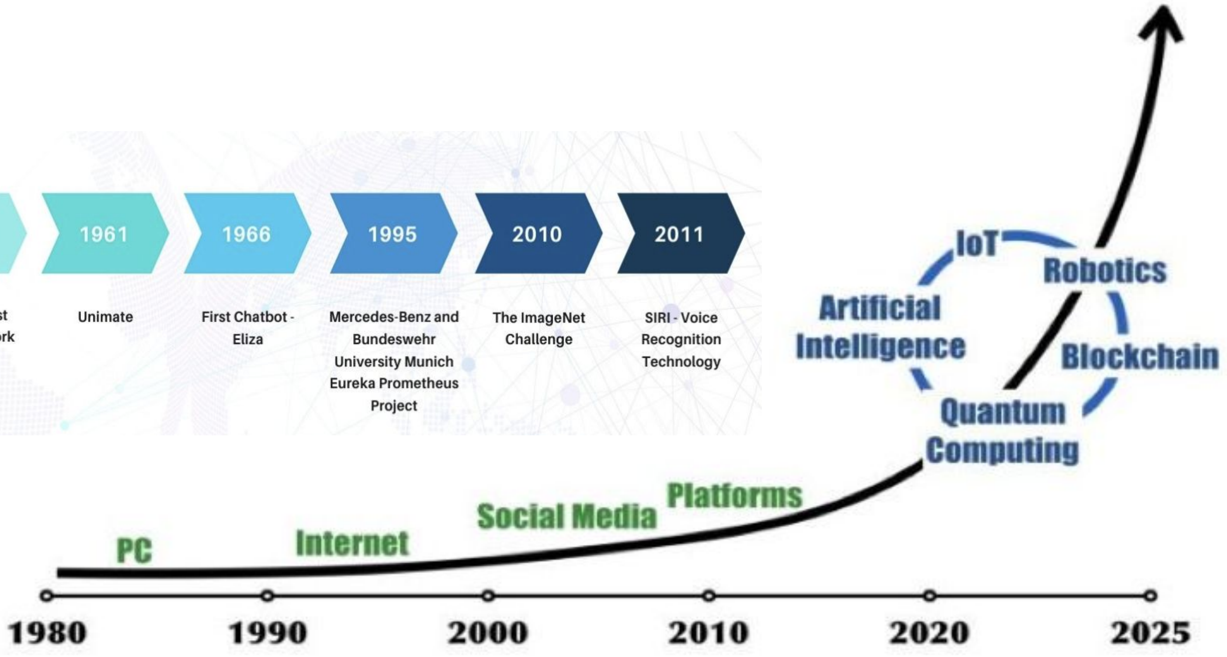
*Machine learning is an application of AI that enables systems to **learn** and **improve** from **experience** without being explicitly programmed.*

Image source Eureka

Evolution of AI



1951
SNARC - First Neural Network
1961
Unimate
1966
First Chatbot - Eliza
1995
Mercedes-Benz and Bundeswehr University Munich Eureka Prometheus Project
2010
The ImageNet Challenge
2011
SIRI - Voice Recognition Technology



Exponential Growth of Technology

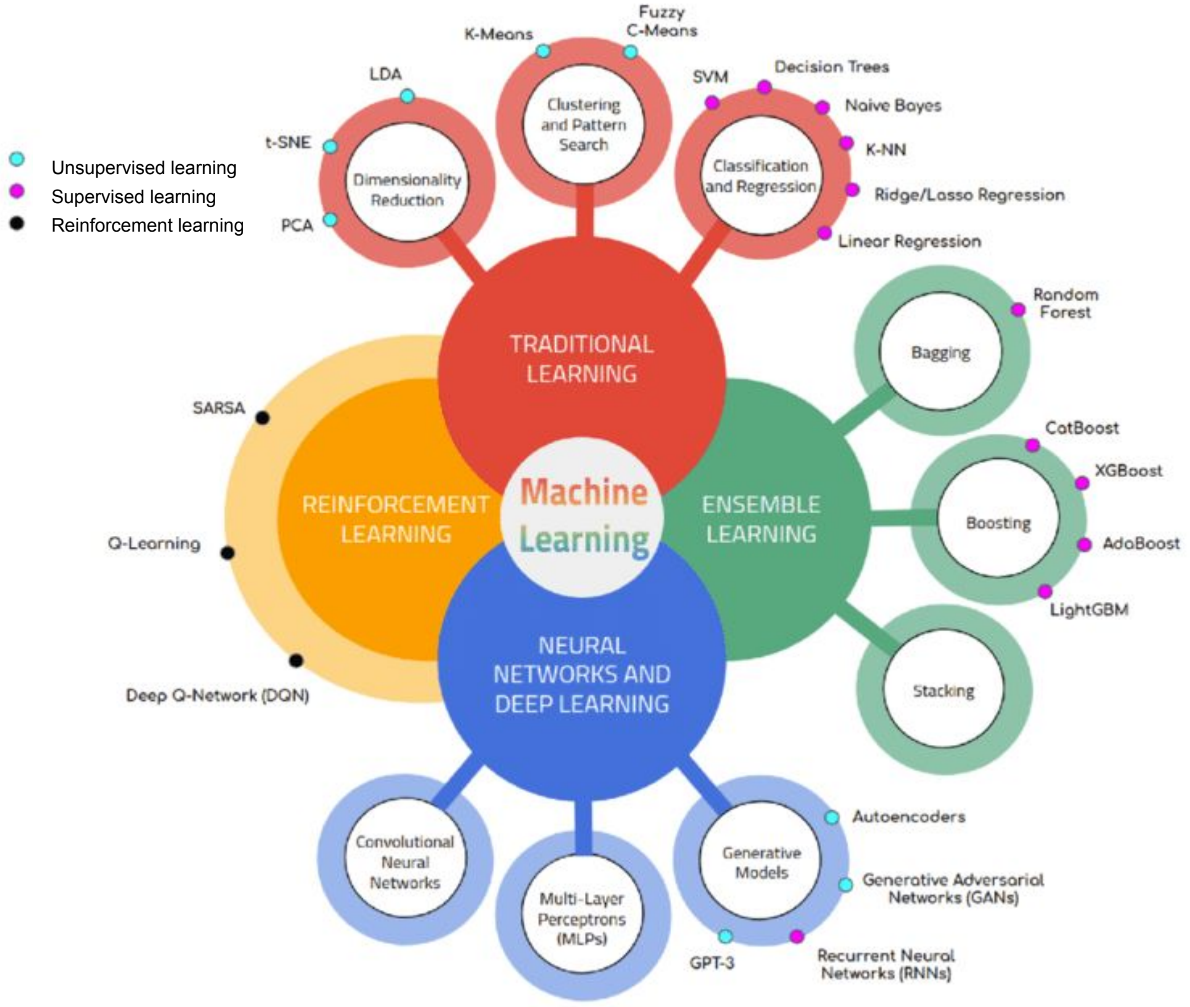
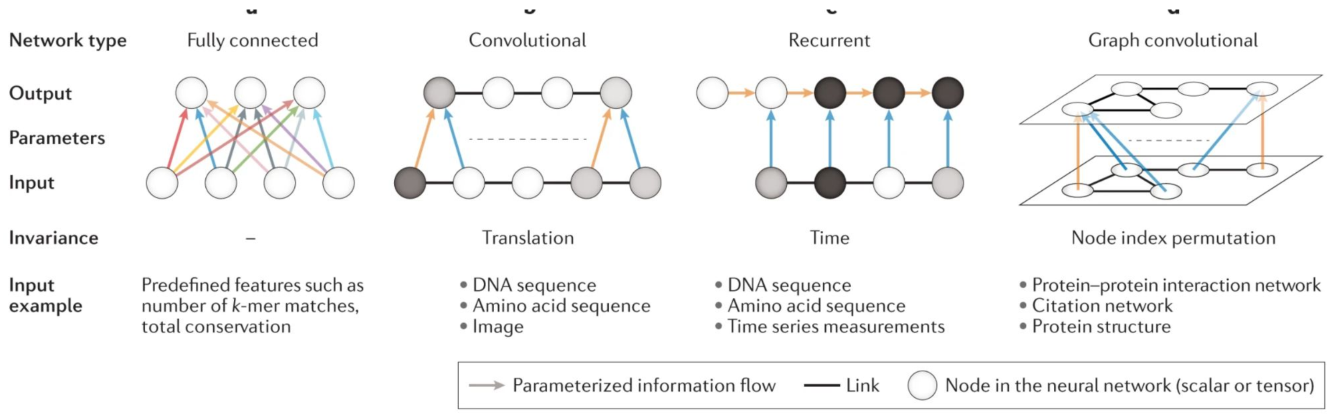


Image source:
DOI:10.1007/s12039-021-01995-2

Deep Learning Applications for Genomics

1. Pattern recognition
2. Predicting biomolecule structures
3. Classification or predictive modeling
4. Image analysis

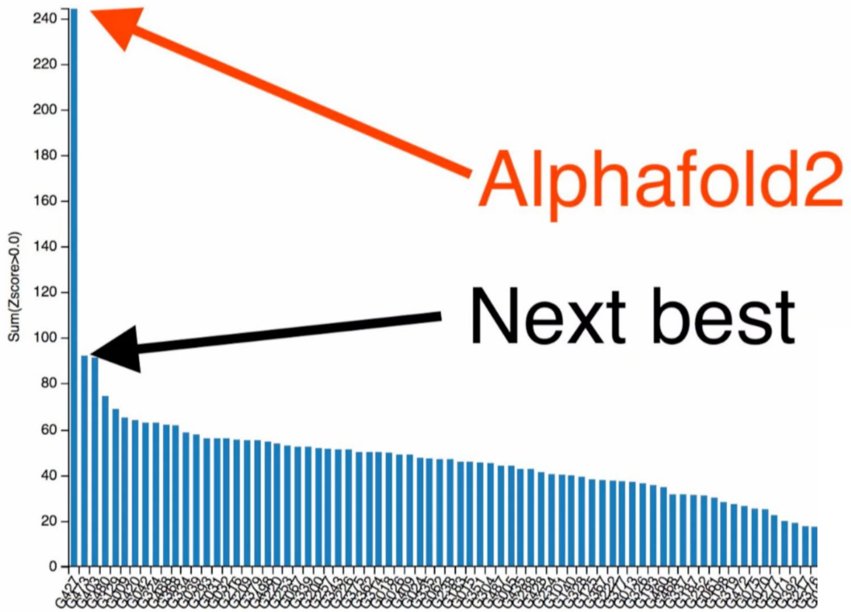


Network type	Fully connected	Convolutional	Recurrent	Graph convolutional
Invariance	–	Translation	Time	Node index permutation
Input example	Predefined features such as number of *k*-mer matches, total conservation	• DNA sequence • Amino acid sequence • Image	• DNA sequence • Amino acid sequence • Time series measurements	• Protein–protein interaction network • Citation network • Protein structure

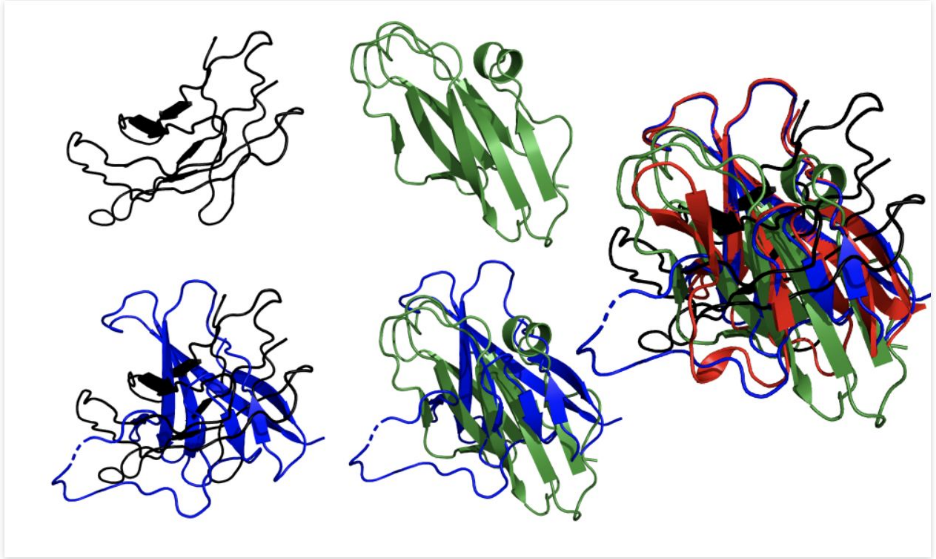
→ Parameterized information flow — Link ○ Node in the neural network (scalar or tensor)

+
Transformers and autoencoders

Eraslan *et al* 2019



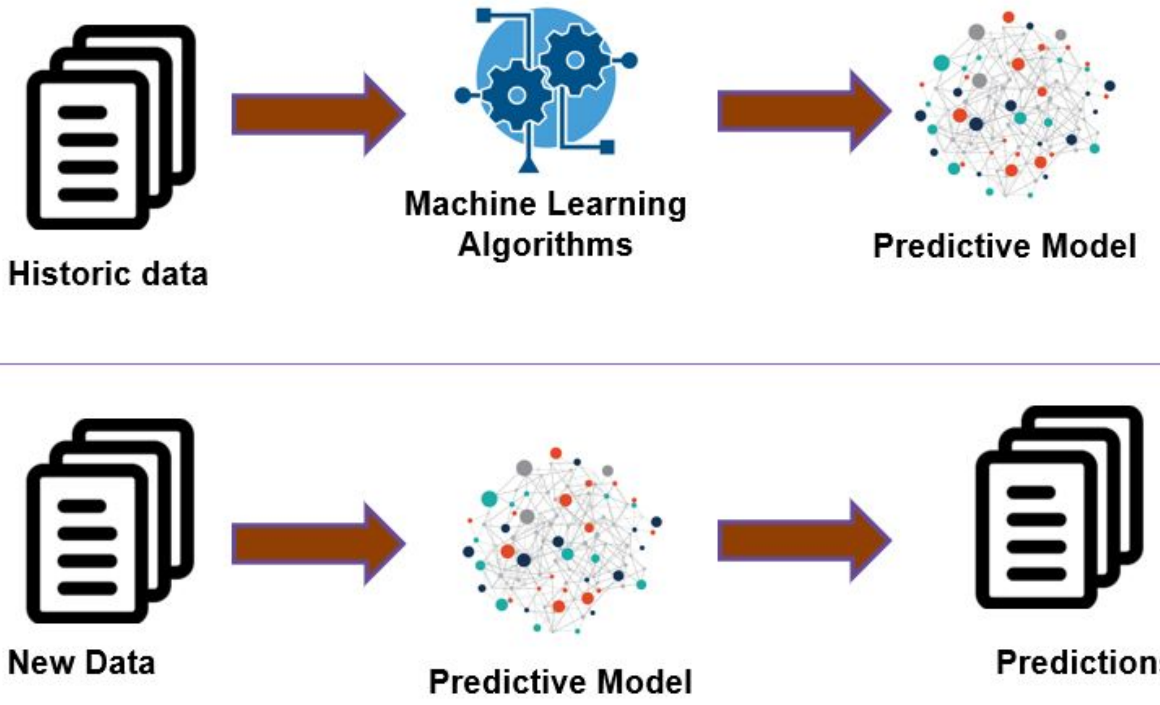
14th CASP Winner



Top: highest-ranked models for the target T1064 submitted by the Zhang (black) and Baker (green) human groups. Bottom: models aligned with the crystal structure. Right: all three models (Zhang, Baker and AlphaFold 2) aligned with the crystal structure. The submissions were obtained from the CASP14 webpage on Tuesday 1st December, 2020.

A typical ML workflow: Supervised Learning

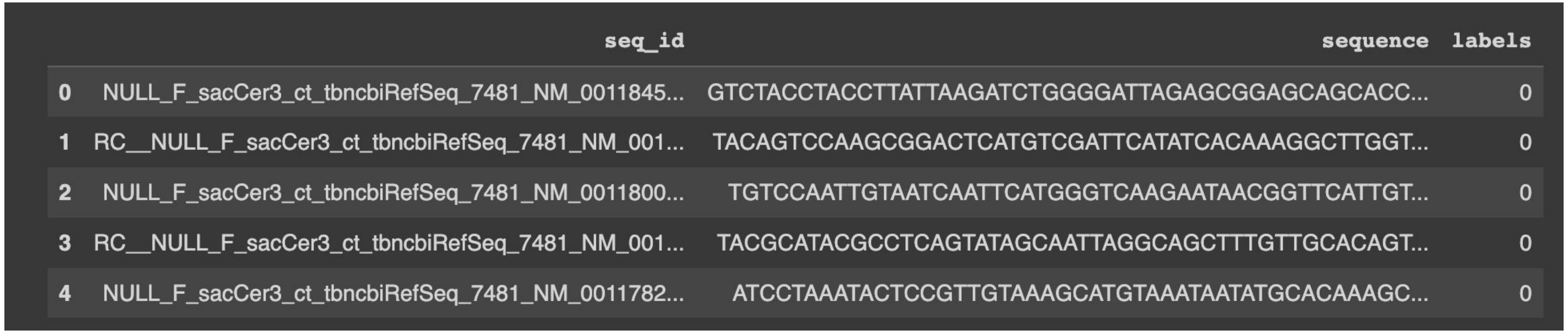
How ML modeling is done?



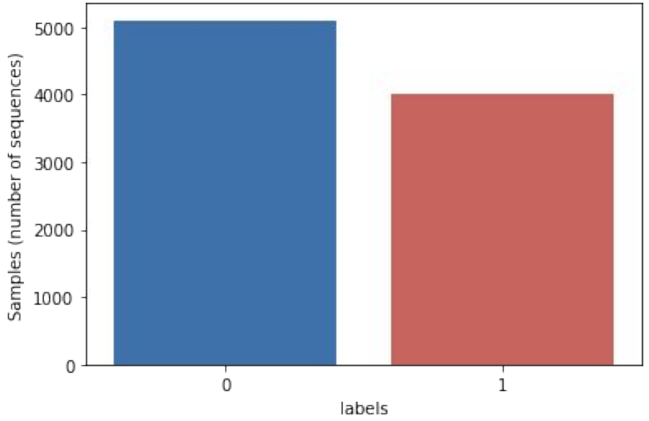
Model training

Scoring

Labelled data for supervised learning



	seq_id	sequence	labels
0	NULL_F_sacCer3_ct_tbncbiRefSeq_7481_NM_0011845...	GTCTACCTACCTTATTAAGATCTGGGGATTAGAGCGGAGCAGCACC...	0
1	RC__NULL_F_sacCer3_ct_tbncbiRefSeq_7481_NM_001...	TACAGTCCAAGCGGACTCATGTCGATTCATATCACAAAGGCTTGGT...	0
2	NULL_F_sacCer3_ct_tbncbiRefSeq_7481_NM_0011800...	TGTCCAATTGTAATCAATTCATGGGTCAAGAATAACGGTTCATTGT...	0
3	RC__NULL_F_sacCer3_ct_tbncbiRefSeq_7481_NM_001...	TACGCATACGCCTCAGTATAGCAATTAGGCAGCTTTGTTGCACAGT...	0
4	NULL_F_sacCer3_ct_tbncbiRefSeq_7481_NM_0011782...	ATCCTAAATACTCCGTTGTAAAGCATGTAAATAATATGCACAAAGC...	0



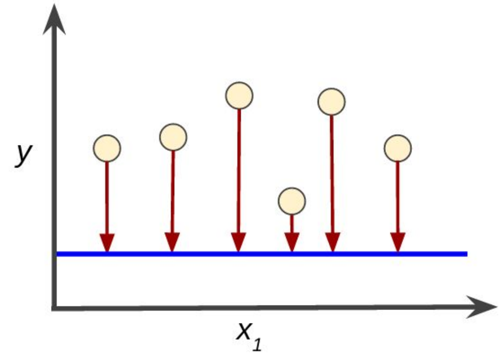
Training and test data: Dividing into two or three parts:

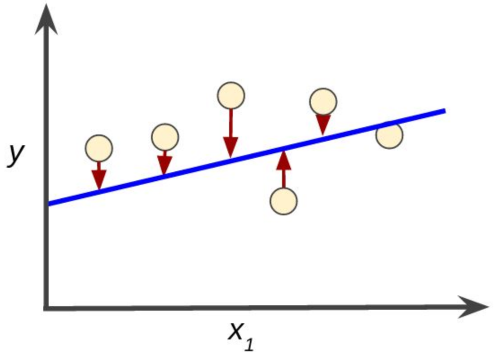
Training data: train the model

Test data: test and iterate until we have the best model

Validation data: validate on a held out unseen data

What happens during training:



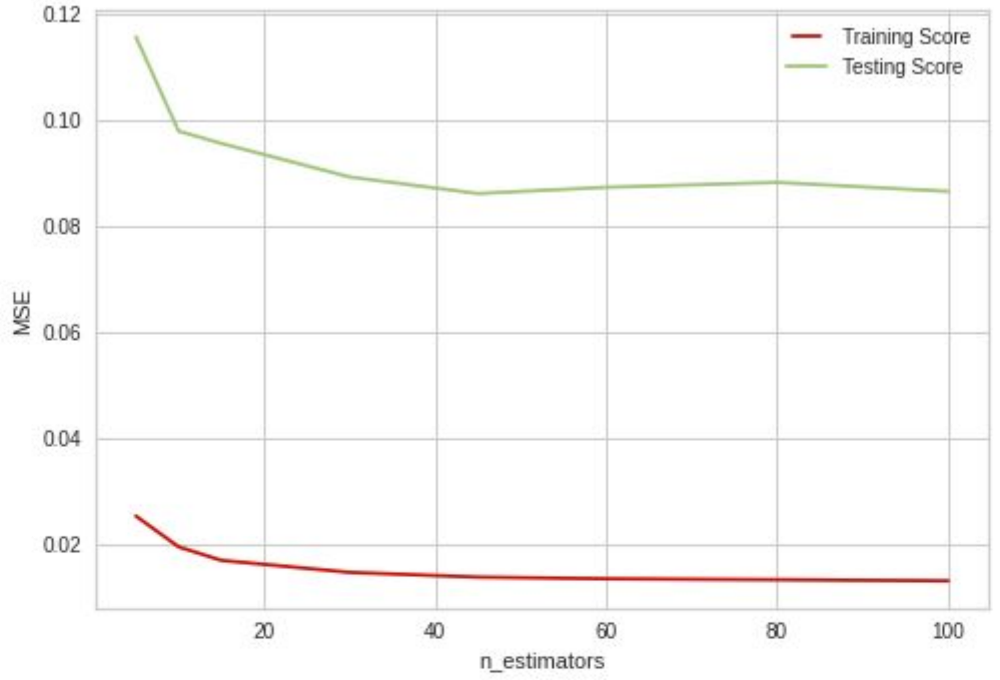


Training a model simply means learning i.e. determining good values (for all the weights and the bias) from labeled examples.

Choosing parameters that minimize the loss

Can we have a direction to go in a parameter space?

Compute the gradient derivative of loss function



https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

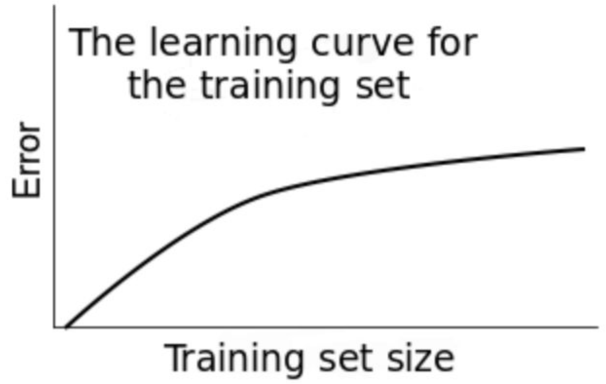
Bias and Variance

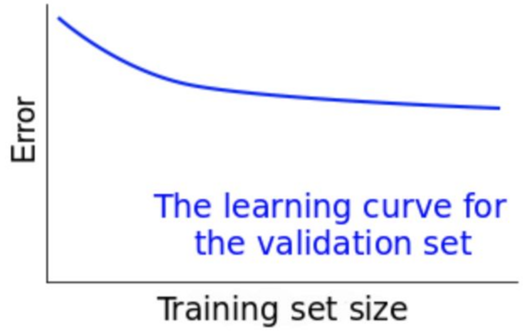
We want to keep error as low as possible

Two major sources of error are bias and variance.

Assumption of supervised learning:

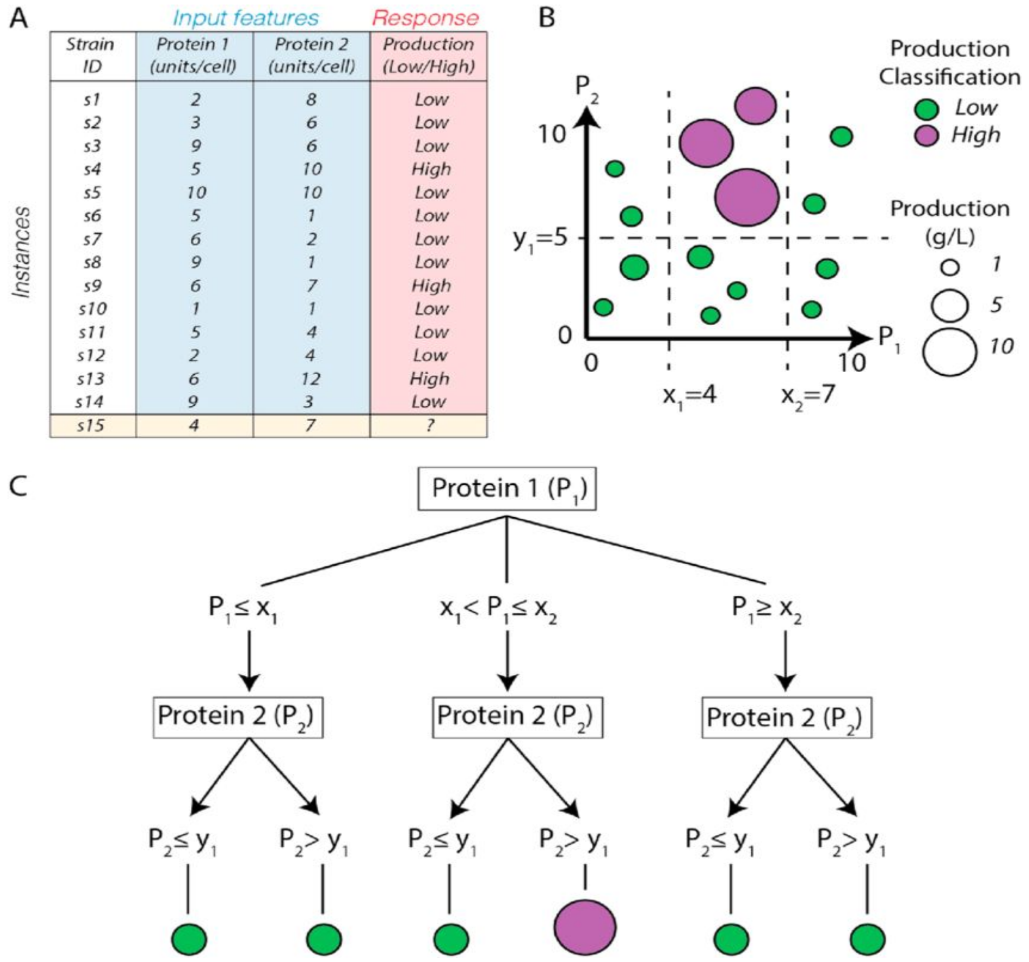
- There is a relationship between the feature(s) and target
- We estimate this relationship using a model (unknown)
- We train different training set to build a model and measure (repeat)
    - The difference between the outcomes of these models describing the relationship between features and target is called “variance”
    - Assumption about the relationship between feature(s) and target is simplified as “bias”





Decision trees

- ➢ The decision tree classifiers organizes a series of test questions and conditions in a tree structure.
- ➢ The root and internal nodes contain attribute test conditions to separate data entries that have different characteristics.
- ➢ All the terminal node is assigned a class label Yes or No.



A

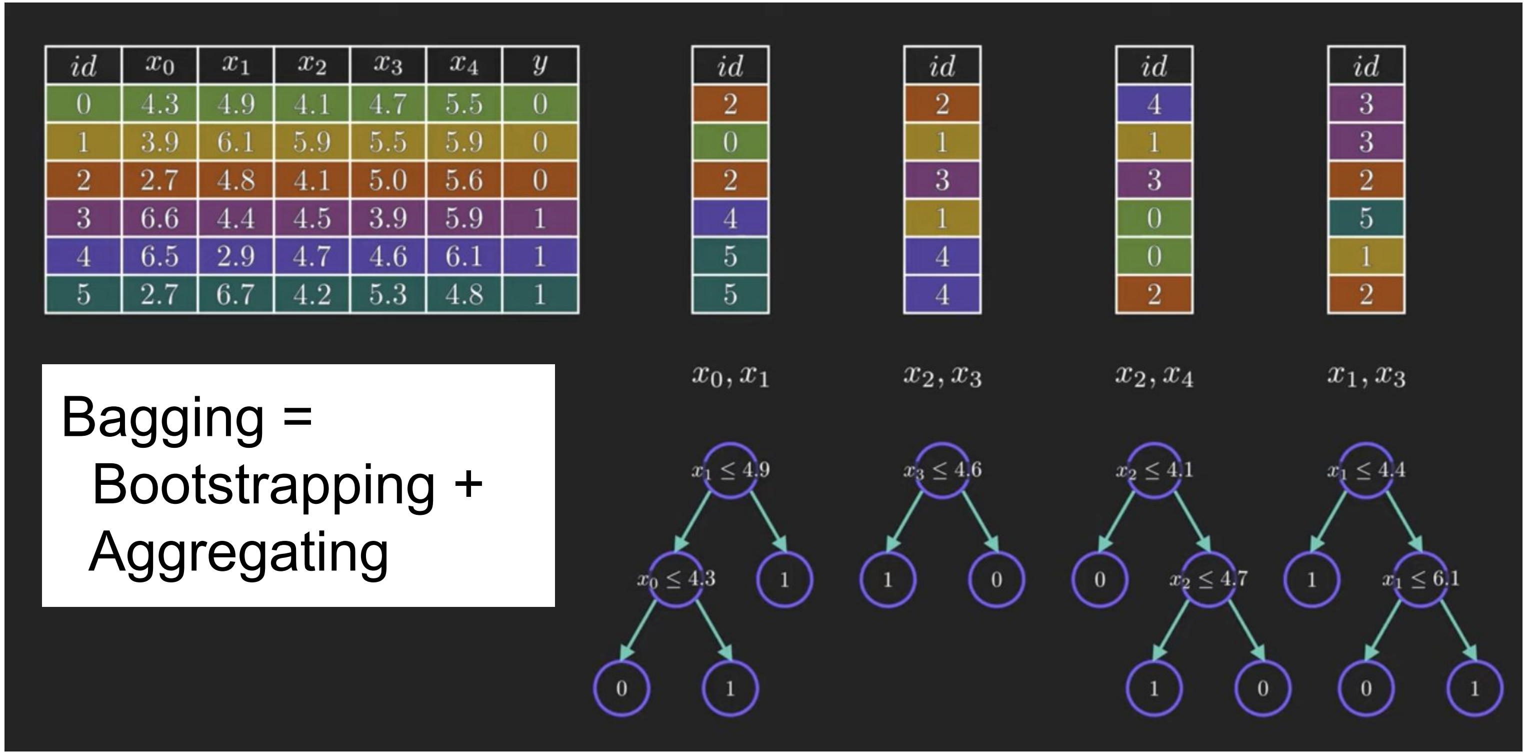
	Input features		Response
Strain ID	Protein 1 (units/cell)	Protein 2 (units/cell)	Production (Low/High)
s1	2	8	Low
s2	3	6	Low
s3	9	6	Low
s4	5	10	High
s5	10	10	Low
s6	5	1	Low
s7	6	2	Low
s8	9	1	Low
s9	6	7	High
s10	1	1	Low
s11	5	4	Low
s12	2	4	Low
s13	6	12	High
s14	9	3	Low
s15	4	7	?

Decision trees pros and cons

- ❖ Scale invariant
- ❖ Robust to irrelevant features
- ❖ Interpretability

- ❖ Prone to overfitting
- ❖ Don’t generalize well
- ❖ Pruning can help alleviate but can not diminish the effects

Random Forest



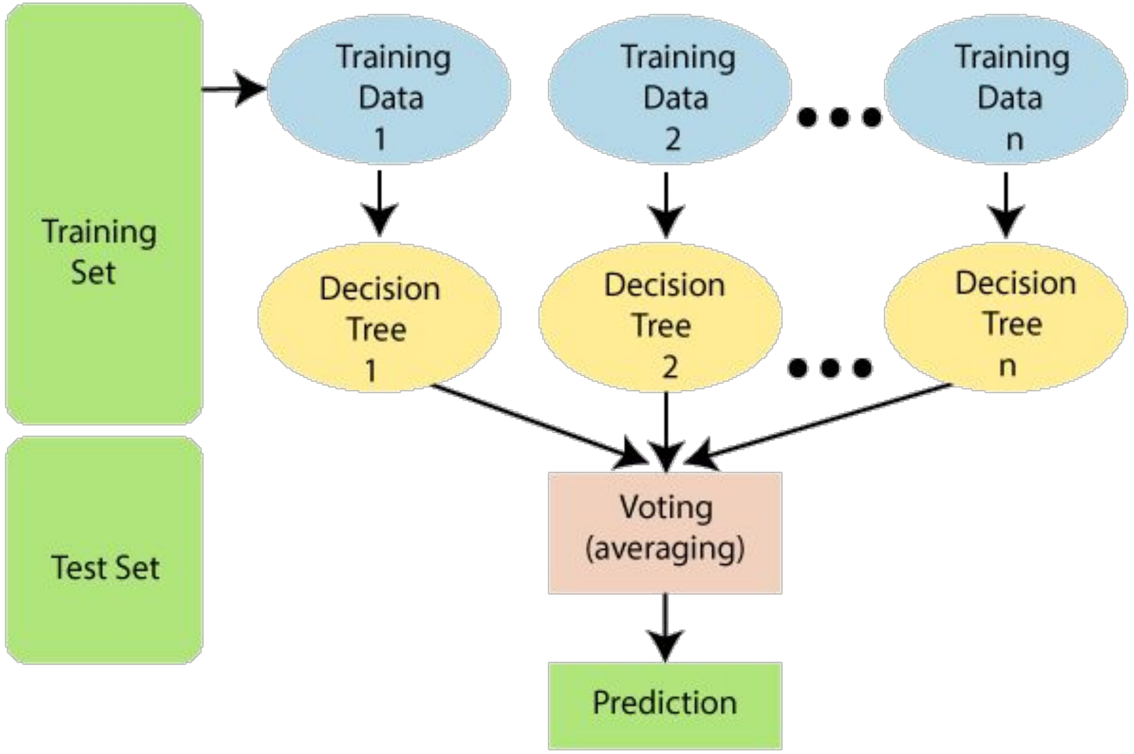
id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1

id	id	id	id
2	2	4	3
0	1	1	3
2	3	3	2
4	1	0	5
5	4	0	1
5	4	2	2

Bagging =
Bootstrapping +
Aggregating

Ref: ML blog

Workflow:



www.javatpoint.com/machine-learning-random-forest-algorithm

The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision.

Decision trees pros and cons

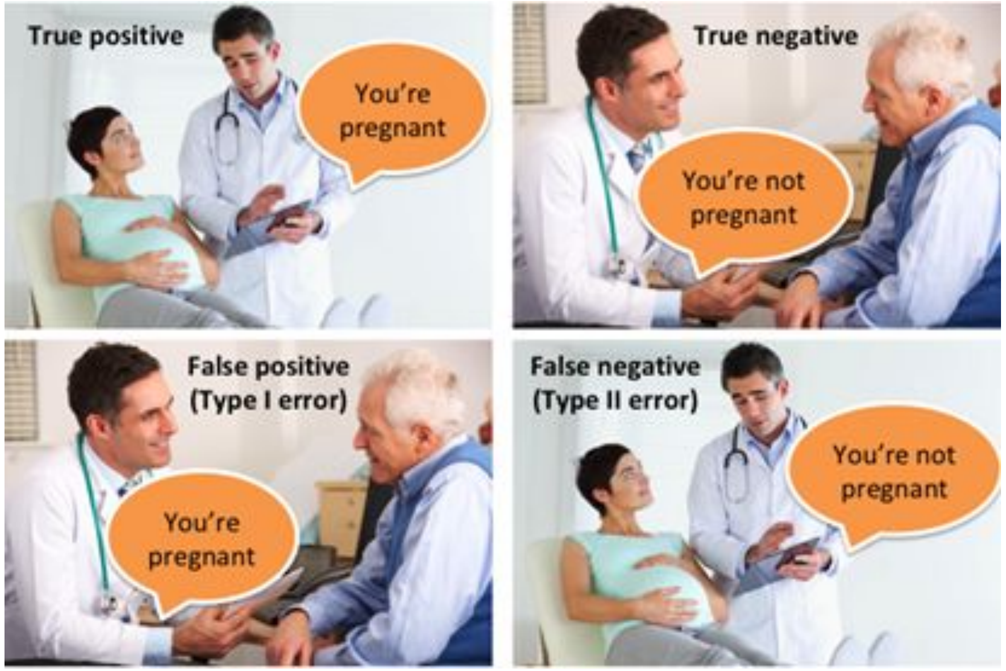
- ❖ Scale invariant
- ❖ Robust to irrelevant features
- ❖ Interpretability

- ❖ Prone to overfitting
- ❖ Don’t generalize well
- ❖ Pruning can help alleviate but can not diminish the effects

RF: Thousands of trees (each tree may overfit but their combined decision is considered by majority votes)

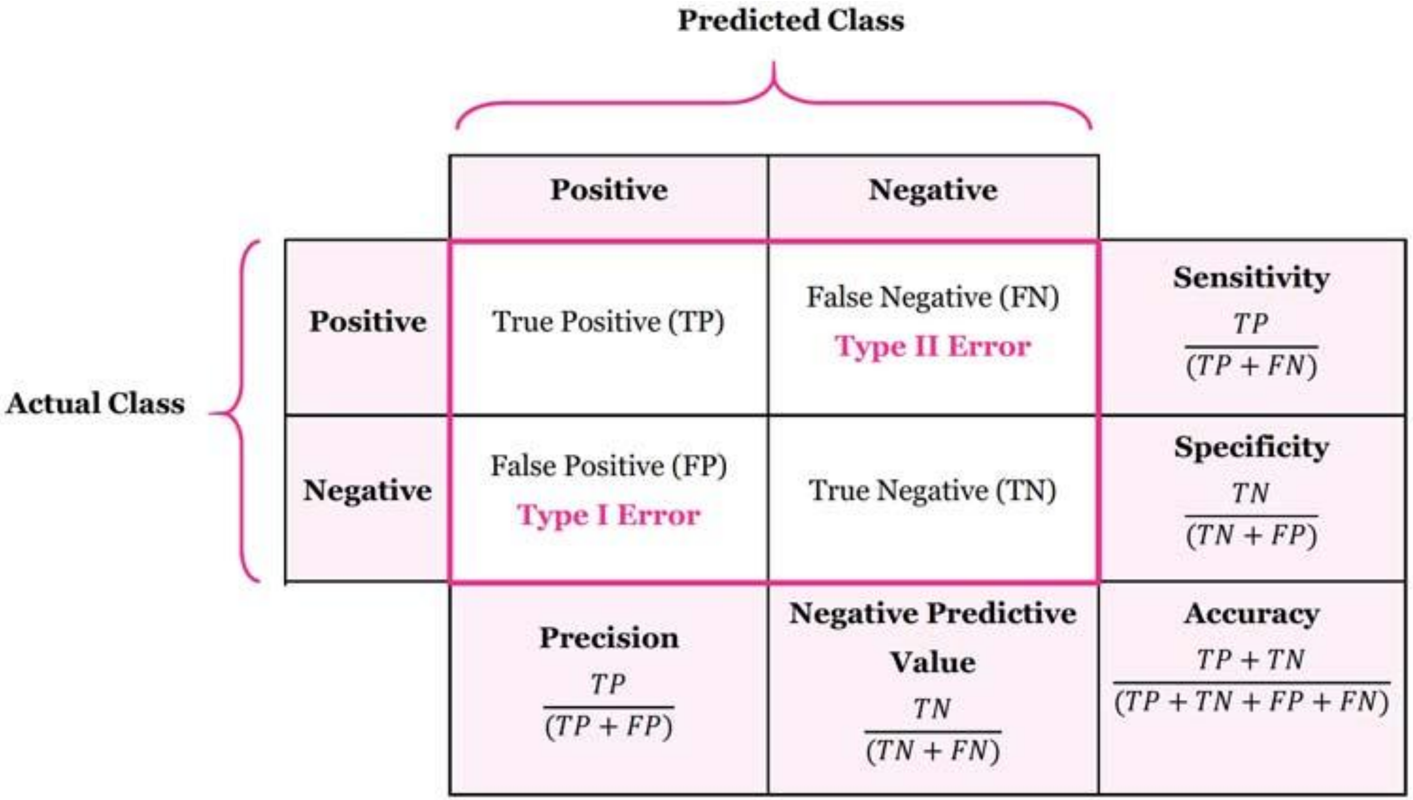
-Allowing variability in both dimensions allows better generalization.

Confusion matrix



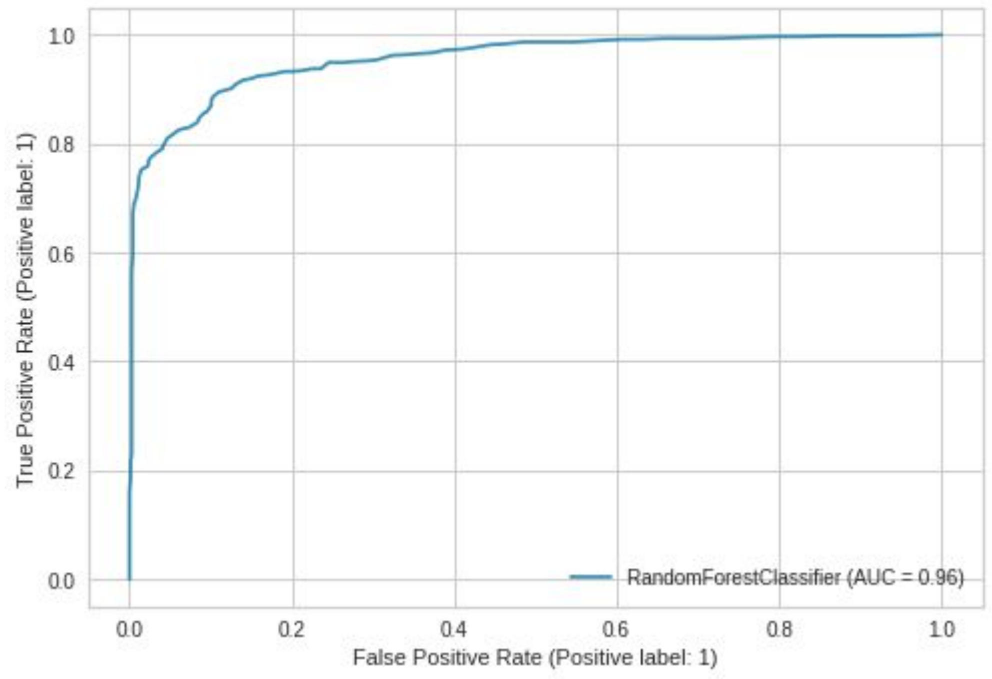
Source: *https://medium.com/@neeraj.kumar.iitg/statistical-performance-measures-12bad66694b7*

Confusion matrix:



		Predicted Class		
		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

ML model performance assessment



Confusion matrix

Accuracy

F1 Score

Precision

Recall

FPP = 1-specificity

TPP=sensitivity

https://aiineverything.blogspot.com/2021/08/misconception--around-roc-auc.html

Preprocessing of Genomic Data

Slides by Naima Vahab

NLP preprocessing of sequence data:

1. Tokenization: Finding K-mers
2. Numerical encoding:
    a. Ordinal Encoding
    b. One-Hot Encoding
3. Transformations
    a. Frequency tables
        i. N-grams
        ii. Term Frequency - Inverse Document Frequency
    b. Embedding Vector
    c. Positional encoding

Terminology used in this document:

Token: smallest unit of processing text data. Also known as k-mer in genomics.

Corpus: reference library or database of all tokens

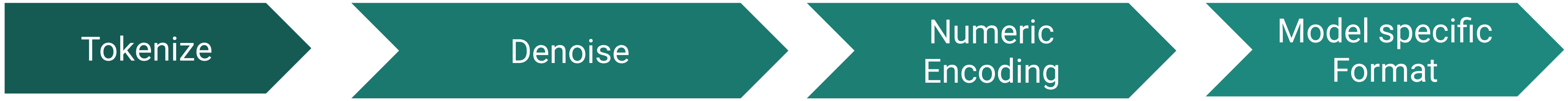
Vector: a set of numbers

Matrix: a two dimensional table of numbers

Stop_words: token with less information content e.g. the, a, an etc in English or known repeat patterns in a genomic sequence

Encoding: converted into a coded form

Workflow : Before applying ML models on genomic sequence, following steps are used to preprocess the data



Convert data into Tokens, also known as k-mers

Remove unimportant features/ tokens

ML algorithms requires data in numbers

According to selected model convert the data into vector/ matrix etc.

Tokenization : Split the genome sequence in arbitrary length

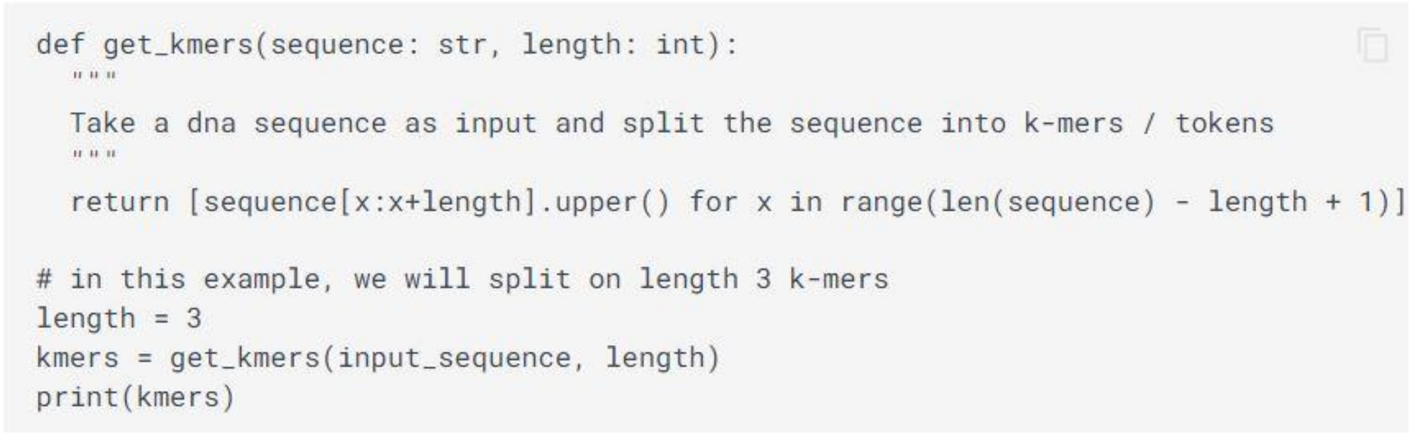
Example:

`INPUT : TAATG…`

`KMER_1: TAA--`

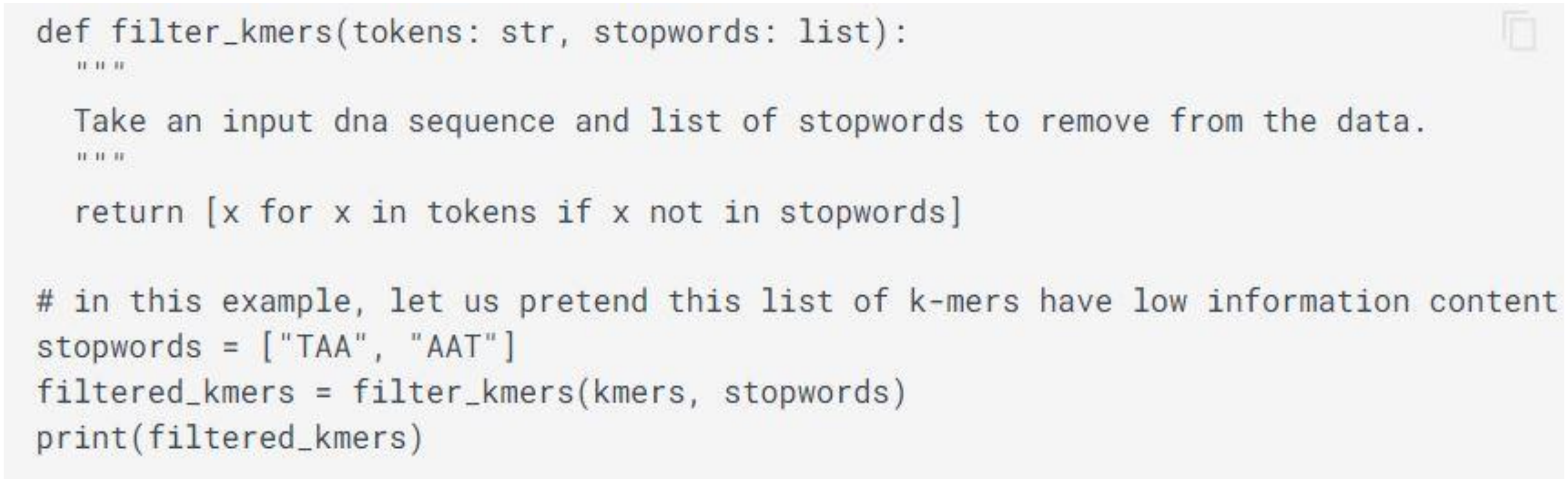
`KMER_2: -AAT-`

`KMER_3: --ATG`



```python
def get_kmers(sequence: str, length: int):
    """
    Take a dna sequence as input and split the sequence into k-mers / tokens
    """
    return [sequence[x:x+length].upper() for x in range(len(sequence) - length + 1)]

# in this example, we will split on length 3 k-mers
length = 3
kmers = get_kmers(input_sequence, length)
print(kmers)
```

Denoise the data : Removing less important features or stop words

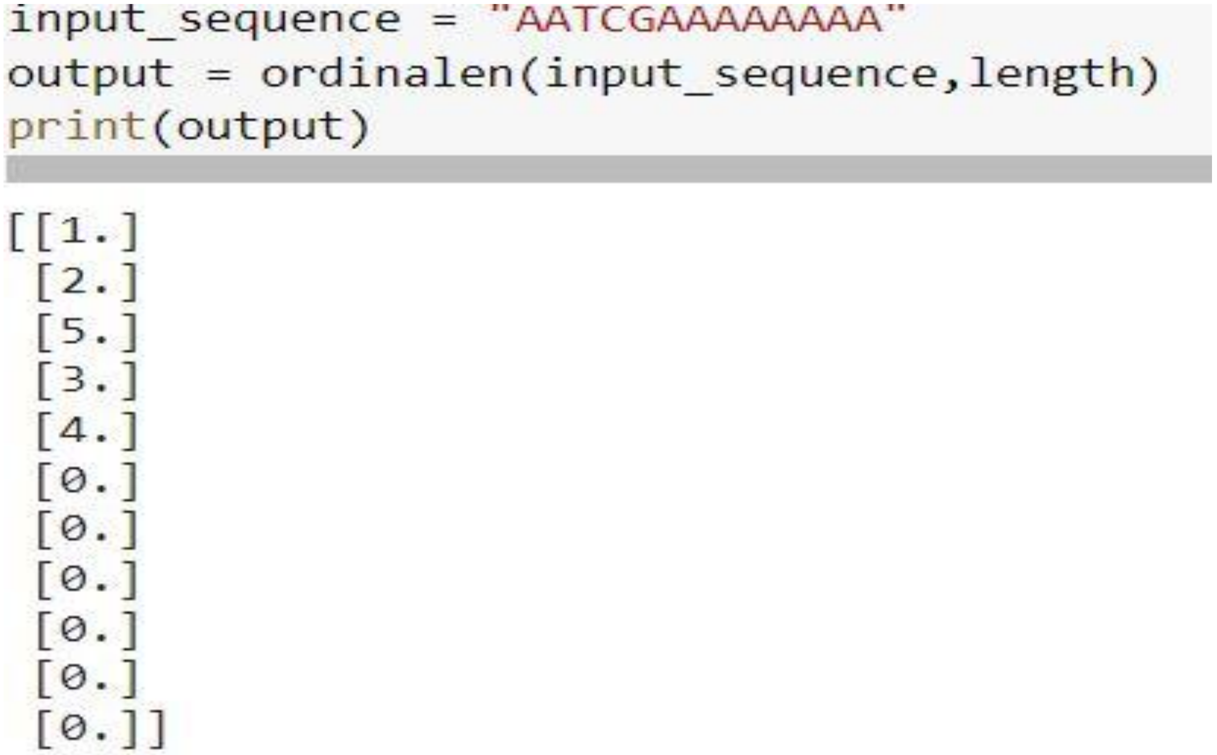


```
def filter_kmers(tokens: str, stopwords: list):
    """
    Take an input dna sequence and list of stopwords to remove from the data.
    """
    return [x for x in tokens if x not in stopwords]

# in this example, let us pretend this list of k-mers have low information content
stopwords = ["TAA", "AAT"]
filtered_kmers = filter_kmers(kmers, stopwords)
print(filtered_kmers)
```

Ordinal encoding : Assigns an integer to each category value

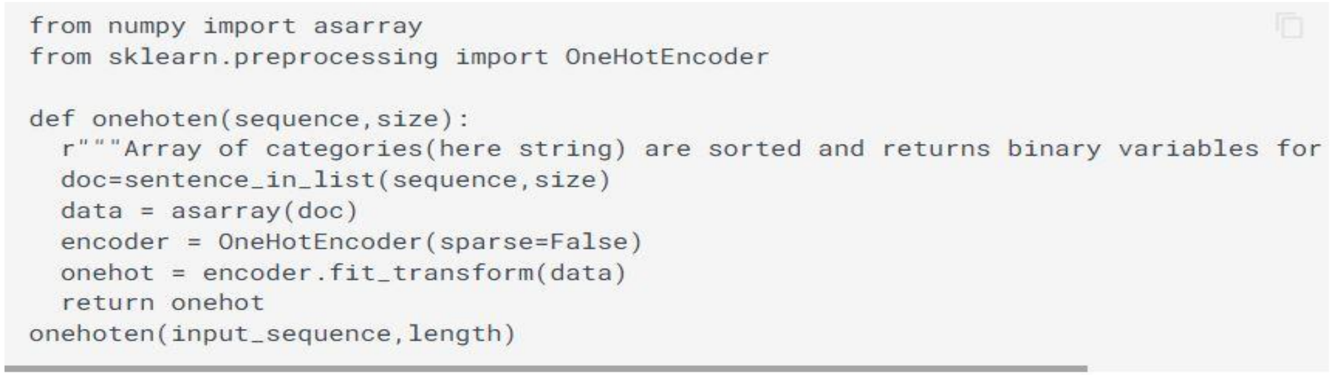
```
from sklearn.preprocessing import OrdinalEncoder

def encode_ordinal(input_sequence: str, length: int):
    """
    Take a list of k-mers and perform ordinal encoding
    """
    tokens = [[x] for x in get_kmers(input_sequence, length)]
    encoder = OrdinalEncoder()
    return encoder.fit_transform(tokens)

ordinal = encode_ordinal(input_sequence, length)
print(ordinal)
```

```
[[5.]
 [0.]
 [1.]
 [7.]
 [4.]
 [3.]
 [2.]
 [6.]]
```

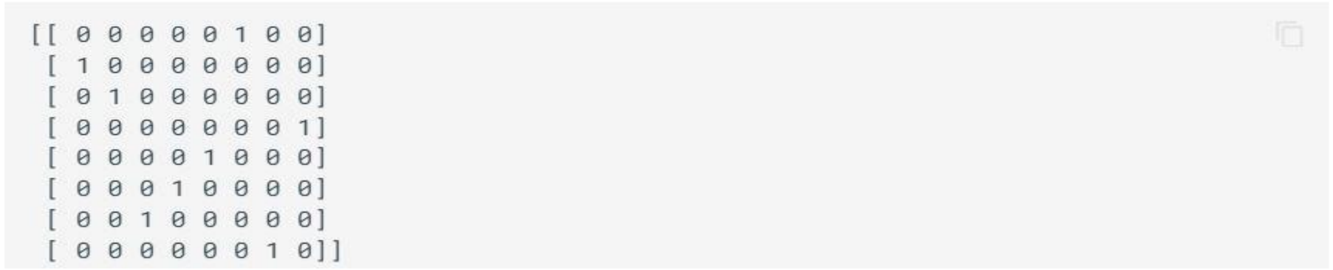

```
input_sequence = "AATCGAAAAAAAA"
output = ordinalen(input_sequence,length)
print(output)
```

```
[[1.]
 [2.]
 [5.]
 [3.]
 [4.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]]
```

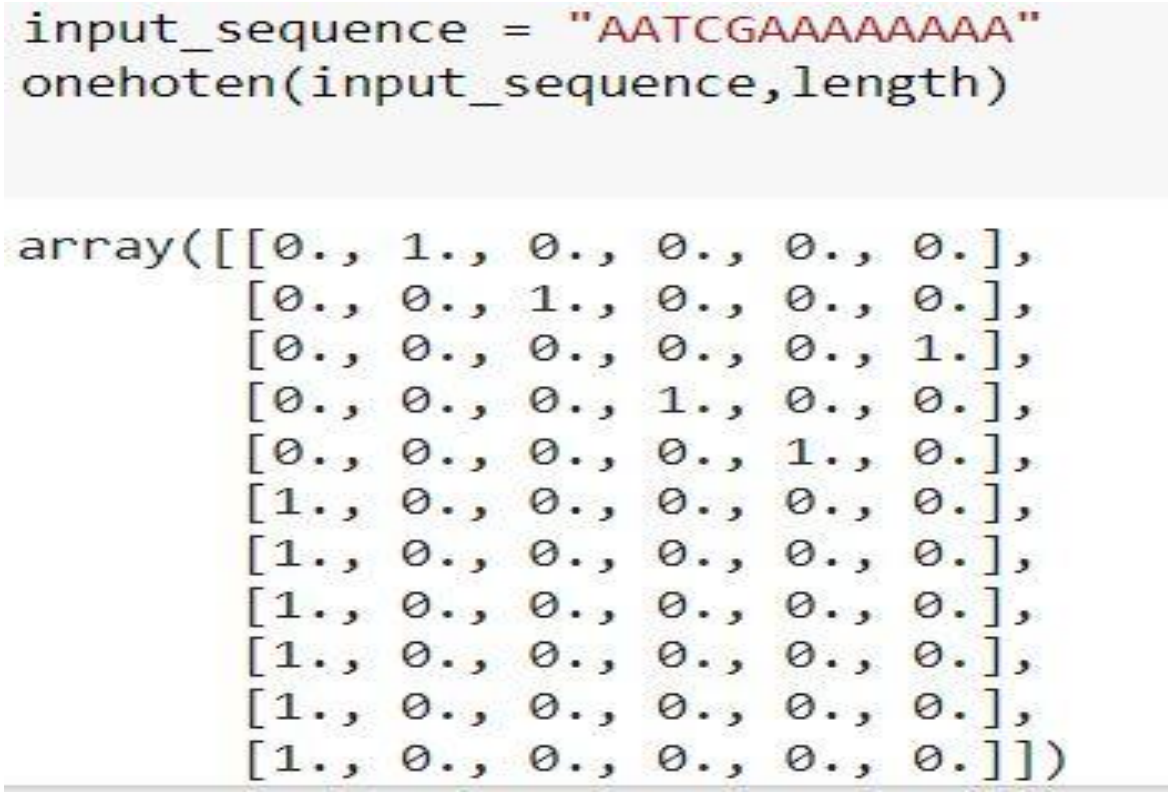
One-Hot Encoding : Each label is represented by binary format



```
from numpy import asarray
from sklearn.preprocessing import OneHotEncoder

def onehoten(sequence,size):
  r"""Array of categories(here string) are sorted and returns binary variables for
  doc=sentence_in_list(sequence,size)
  data = asarray(doc)
  encoder = OneHotEncoder(sparse=False)
  onehot = encoder.fit_transform(data)
  return onehot
onehoten(input_sequence,length)
```



```
[[ 0  0  0  0  0  1  0  0]
 [ 1  0  0  0  0  0  0  0]
 [ 0  1  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  1]
 [ 0  0  0  0  1  0  0  0]
 [ 0  0  0  1  0  0  0  0]
 [ 0  0  1  0  0  0  0  0]
 [ 0  0  0  0  0  0  1  0]]
```



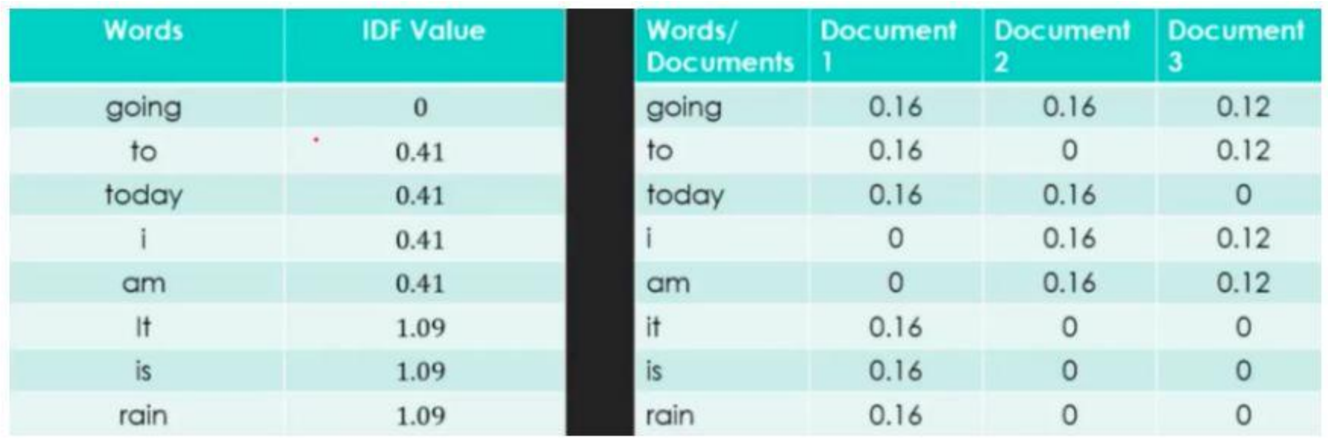
```
input_sequence = "AATCGAAAAAAA"
onehoten(input_sequence,length)
```

```
array([[0., 1., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 0., 1.],
       [0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0.],
       [1., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0.]])
```

TF-IDF (Term Frequency-Inverse Document Frequency) : Multiplying TF and IDF implies a weight to term which gives how important is a word in the document.

TF = (Number of repetitions of word in a document) / (# of words in a document)

IDF =Log[(Number of documents) / (Number of documents containing the word)]



Words	IDF Value
going	0
to	0.41
today	0.41
i	0.41
am	0.41
It	1.09
is	1.09
rain	1.09

Words/ Documents	Document 1	Document 2	Document 3
going	0.16	0.16	0.12
to	0.16	0	0.12
today	0.16	0.16	0
i	0	0.16	0.12
am	0	0.16	0.12
it	0.16	0	0
is	0.16	0	0
rain	0.16	0	0

IDF Value and TF value of 3 documents.

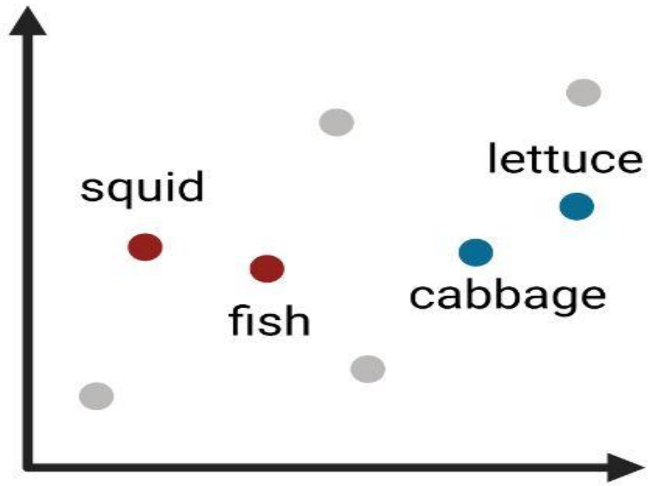
```
from sklearn.feature_extraction.text import TfidfVectorizer

def tfidf(input_sequence: str, length: int):
  """
  Take a dna sequence and k-mer size as input,
  output a matrix of TF-IDF features.
  """
  data = [" ".join(get_kmers(input_sequence, length))]
  tfidf = TfidfVectorizer()
  tfidf = tfidf.fit_transform(data)
  return tfidf.toarray()

tfidf_matrix = tfidf(input_sequence,length)
print(tfidf_matrix)
```

```
input_sequence = "AATCGAAAAAAAA"
output = tfidf(input_sequence,length)
print(output)
```

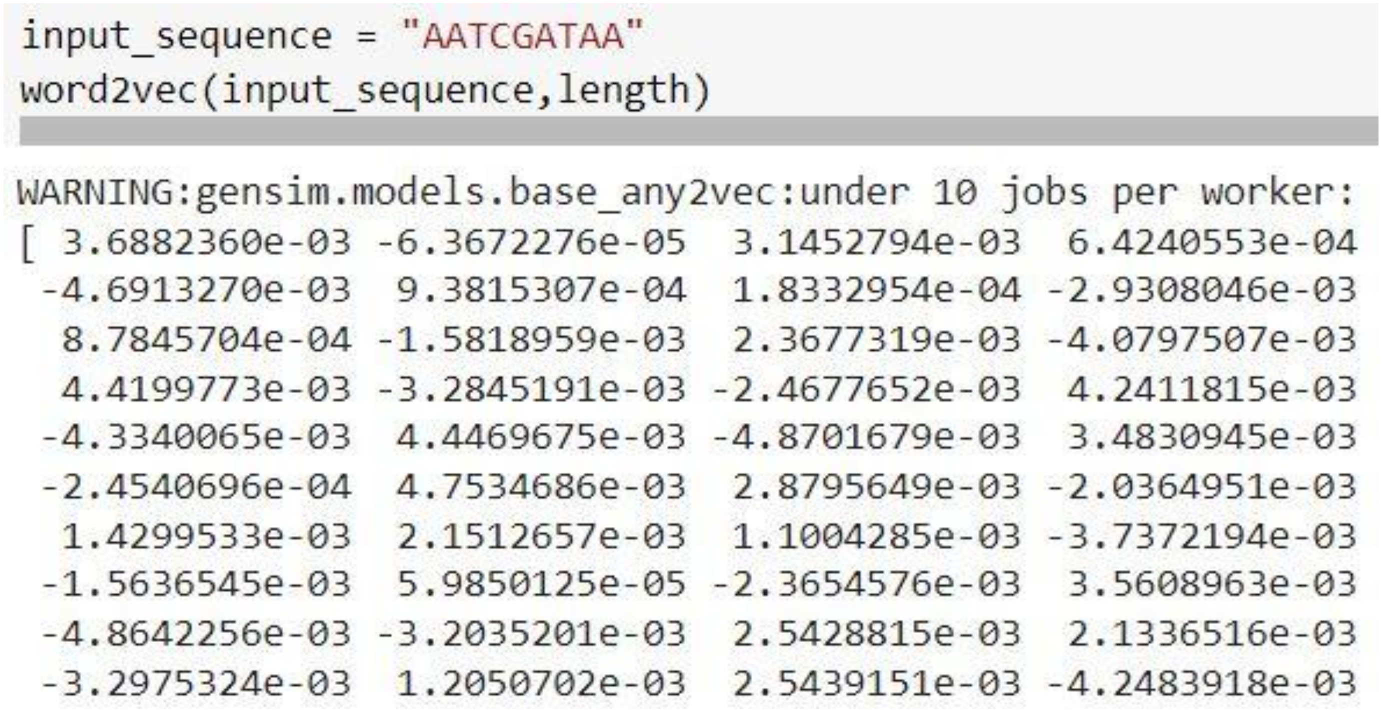
```
tf-idf values in matrix form:
[[0.93704257 0.15617376 0.15617376 0.15617376 0.15617376 0.15617376]]
```

Embedding Vector : Taking a single sequence and projecting into the embedding returns a unique vector for each sequence.

Can use libraries like word2vec, dna2vec, GloVe etc.



```
import gensim
from gensim.models import Word2Vec

def create_word2vec(input_sequence: str, length: int):
  """
  Input a list of tokens to generate an embedding of word vectors
  """
  data = get_kmers(input_sequence, length)
  return Word2Vec([data], min_count=1)

def map_token(input_sequence: str, model: gensim.models.base_any2vec):
  """
  Input a token and embedding and map the token onto the embedding
  """
  return model[input_sequence]

embedding = create_word2vec(input_sequence, length)
vector = map_token('TAA', embedding)
print(vector)
```



```
input_sequence = "AATCGATAA"
word2vec(input_sequence,length)
```

```
WARNING:gensim.models.base_any2vec:under 10 jobs per worker:
[ 3.6882360e-03 -6.3672276e-05  3.1452794e-03  6.4240553e-04
 -4.6913270e-03  9.3815307e-04  1.8332954e-04 -2.9308046e-03
  8.7845704e-04 -1.5818959e-03  2.3677319e-03 -4.0797507e-03
  4.4199773e-03 -3.2845191e-03 -2.4677652e-03  4.2411815e-03
 -4.3340065e-03  4.4469675e-03 -4.8701679e-03  3.4830945e-03
 -2.4540696e-04  4.7534686e-03  2.8795649e-03 -2.0364951e-03
  1.4299533e-03  2.1512657e-03  1.1004285e-03 -3.7372194e-03
 -1.5636545e-03  5.9850125e-05 -2.3654576e-03  3.5608963e-03
 -4.8642256e-03 -3.2035201e-03  2.5428815e-03  2.1336516e-03
 -3.2975324e-03  1.2050702e-03  2.5439151e-03 -4.2483918e-03
```

Models in Summary

Input Sequence : AATAAGTGC

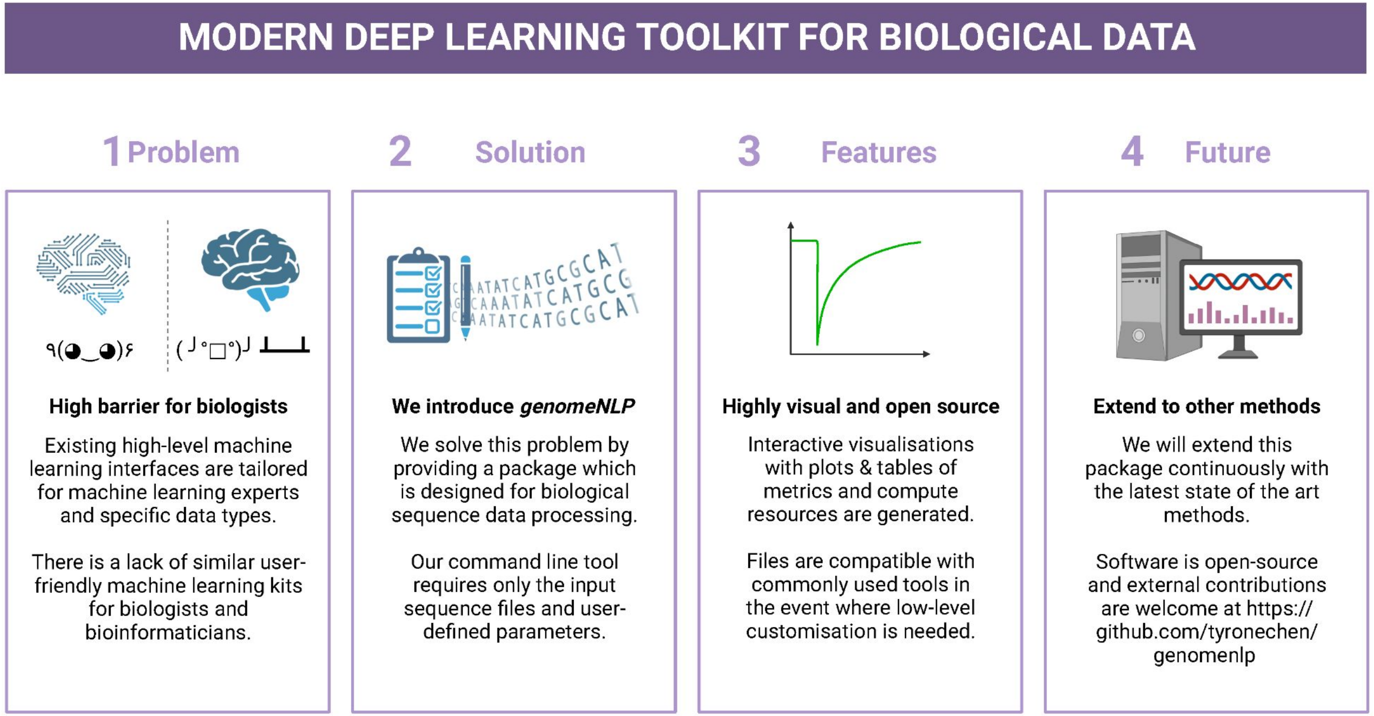
Ordinal Encoding : [[1],[2],[0],[3]]

One-Hot Encoding : [[0,1,0,1],[0,0,1,0],[1,0,0,0],[0,0,0,1]]

TF-IDF : [[0.93704257 0.15617376 0.15617376 0.15617376 0.15617376 0.15617376]]

Embedding Vector : [3.6882360e-03 -6.3672276e-05 3.1452794e-03 6.4240553e-04
-4.6913270e-03 9.3815307e-04 1.8332954e-04 -2.9308046e-03
8.7845704e-04 -1.5818959e-03 2.3677319e-03 -4.0797507e-03
4.4199773e-03 -3.2845191e-03 -2.4677652e-03 4.2411815e-03]

Introducing **genomeNLP** by Tyrone Chen et al 2022-23



MODERN DEEP LEARNING TOOLKIT FOR BIOLOGICAL DATA

1 Problem

High barrier for biologists

Existing high-level machine learning interfaces are tailored for machine learning experts and specific data types.

There is a lack of similar user-friendly machine learning kits for biologists and bioinformaticians.

2 Solution

We introduce *genomeNLP*

We solve this problem by providing a package which is designed for biological sequence data processing.

Our command line tool requires only the input sequence files and user-defined parameters.

3 Features

Highly visual and open source

Interactive visualisations with plots & tables of metrics and compute resources are generated.

Files are compatible with commonly used tools in the event where low-level customisation is needed.

4 Future

Extend to other methods

We will extend this package continuously with the latest state of the art methods.

Software is open-source and external contributions are welcome at https://github.com/tyronechen/genomenlp

Thank you!

Next:

- Hands-on exercise
- Login to Google colaboratory https://colab.research.google.com

Classification (sqrt(p)) vs regression (p/3)

1. Compute the accuracy on the ith training set (80-20 split)
2. Compute the accuracy on the jth feature (permuted)
3. Subtract the acc of permuted training set from that of the unmodified training set. The difference will be higher for more important features.
4. Average over all training sets

Text representations are mostly used in,

1. Machine Translation — Automatically translating text from one language to another.
2. Document clustering — Grouping text documents based on the structural and/or semantic similarity.
3. Topic detection — Identifying the topic of a large text corpus.
4. Text summarization.
5. Question Answering.

6.

NLP preprocessing of sequence data:

1. Tokenization: Finding K-mers
2. Numerical encoding:
    a. Ordinal Encoding
    b. One-Hot Encoding
3. Transformations
    a. Frequency tables
        i. N-grams
        ii. Term Frequency - Inverse Document Frequency
    b. Embedding Vector
    c. Positional encoding

Some of the text representations are one-hot encoding, n-gram model, Bag-of-Words model and neural word embedding.

In the bag of words model text or the documents are represented by modeling a *bag* (unordered collection) of words. This *bag* of words does not count the positioning, grammar or structure of the words in the text. It just count the frequencies of words in the target text and put that words in to a *bag*. The frequencies of words appearing in a text (sentence or document) is the feature that is used on bag-of-words model.