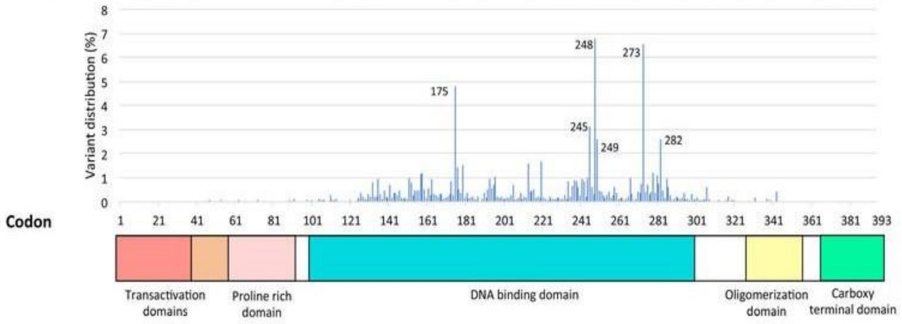
Cancer Classification via Mutations in TP53 Gene

Andrea Puccetti
Claudia Ruggiero
Carlo Conte

Data Mining - A.Y. 2021

TP53 gene role

- Tumoral cells proliferate in a uncontrolled way
- Tp53 gene encodes for protein P53, also known as “Guardian of the Genome”
- Such protein indeed acts as tumour suppressor, expressed when DNA damages and genome mutations occur, to block cells proliferation until the damage is repaired.
- Mutations in TP53 gene itself compromises its functioning, leading to tumorigenesis



Problem : Cancer classification

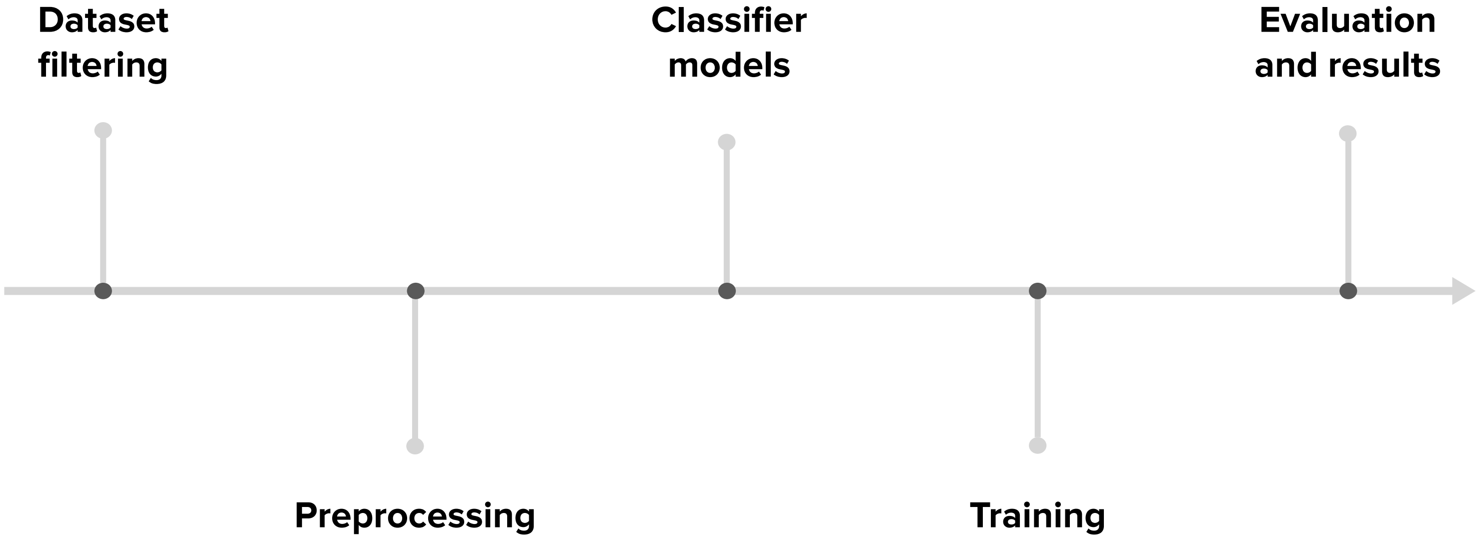
TASK: prediction over 5 categories of cancer type generated basing on TP53 mutations in patients, comparing deep learning techniques with standard approaches

DATASET: the dataset includes informations about patients carrying a malignant mutation in some TP53 codon

APPROACHES:

- Different Neural Network Models (Feed-Forward NN, CNN, LSTM)
- Random Forest classifier
- SVM classifier

Workflow



Dataset filtering

Preprocessing

Classifier models

Training

Evaluation and results

1

PREPROCESSING

Preprocessing

- Filtering original dataset and creation of a new version
  - **selected mutations** : nucleotide changes (A, T, C, G) involving a single TP53 codon in a patient
  - **dataset entry:**

    ID || *codon* || *mutant codon* || *amino acid* || *mutant amino acid* || *type of event* || *complexity*

- Multiple input types considered
- One-Hot-Encoding
- Balancing (up-sampling)

1.1 INPUT TYPES

INPUTS: Simple and Extended inputs

Type 0:

original codon	*mutant codon*
CTT	**CGT**

Type 1:

original codon	*mutant codon*	*binary position*	*amino acid*	*mutant amino acid*	*type of event*	*complexity*
CTT	**CGT**	**194**	**Leu**	**Arg**	**Tv**	**SM**

Alignment problem

Sequence alignment = arrangement of DNA / RNA / proteins sequences to identify regions of similarity

- Required task for most bioinformatics problems
- Hard to predict if similarity relationships between sequences of amino acids are ascribable to a common evolutionary origin
- In general, homologous sequences share similar functionalities

Our approximation:

- Given mutant codon **C**, its position, and a WINDOW SIZE ***n***
- Take the sequence ***S*** of left and right ***n*** - 1 adjacent codons surrounding ***C***
- Consider all the possible subsequences of ***S*** with length ***n***
- **We expect that same subsequences in different patients most likely will generate the same disease**

Subsequences idea and blacklist

AAAACCTACCAGTGCAGCTACGGTTTC

AAAACCTACCAGTGC

TGCAGCTACGGTTTC

ACCTACCAGTGCAGC

CAGTGCAGCTACGGT

TACCAGTGCAGCTAC

Assumption: subsequences that appear more frequently characterize some disease

- Characterizing subsequences for all diseases are included in a **BLACKLIST**
- The frequence threshold for inclusion in blacklist is specific for each disease
- Only entries containing characterizing subsequences are selected for processing

NEW INPUT TYPE *:* *left-adjacent codons || mutant codon || right-adjacent codons*

INPUTS: Sequence inputs

Type 2:

sequence
S

Type 3:

sequence	*amino acid*	*mutant amino acid*	*type of event*	*complexity*
S	**Leu**	**Arg**	**Tv**	**SM**

Type 4:

nucleotide_0

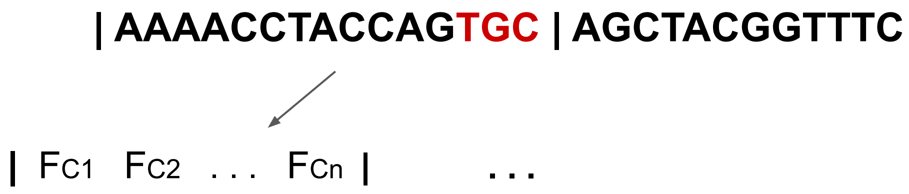
⋮

nucleotide_m

m = # of nucleotides in a sequence

S = AAAACCTACCAGTGCAGCTACGGTTTC

Another idea: frequency



| AAAACCTACCAGTGC | AGCTACGGTTTC

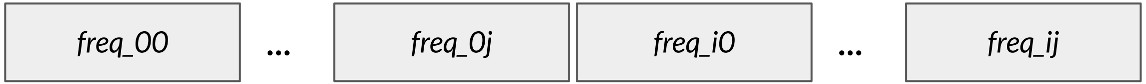
| F_{C1} F_{C2} . . . F_{Cn} | . . .

Assumption: Instead of using a one-hot encoding, encode each subsequence in an array of length #categories. Put in position ***i*** the number of times the subsequence ***s*** appears in category ***Ci***.

- The size of the input will be (**#categories**, **#subsequences**), reducing it up to a 75% compared to other types of input
- It gives a proportional weight to each category for every subsequence

Frequency inputs

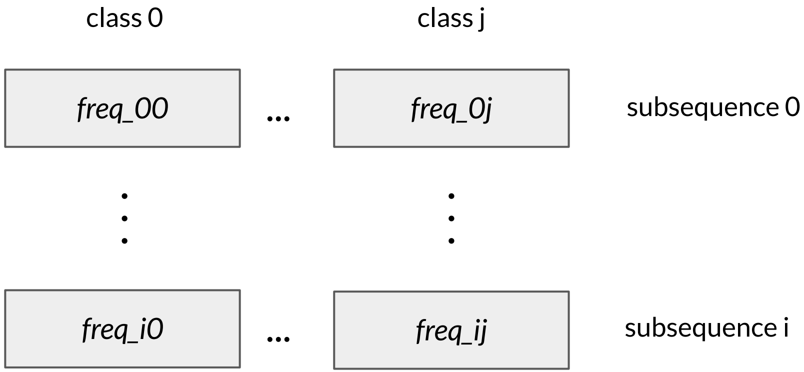
Type 5:



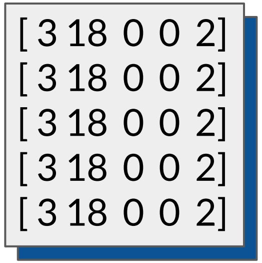
freq_00 ... *freq_0j* *freq_i0* ... *freq_ij*

- i is the i-th subsequence of length *n*
- j is the disease

Type 6:



class 0		class j	
freq_00	...	*freq_0j*	subsequence 0
⋮		⋮	
freq_i0	...	*freq_ij*	subsequence i

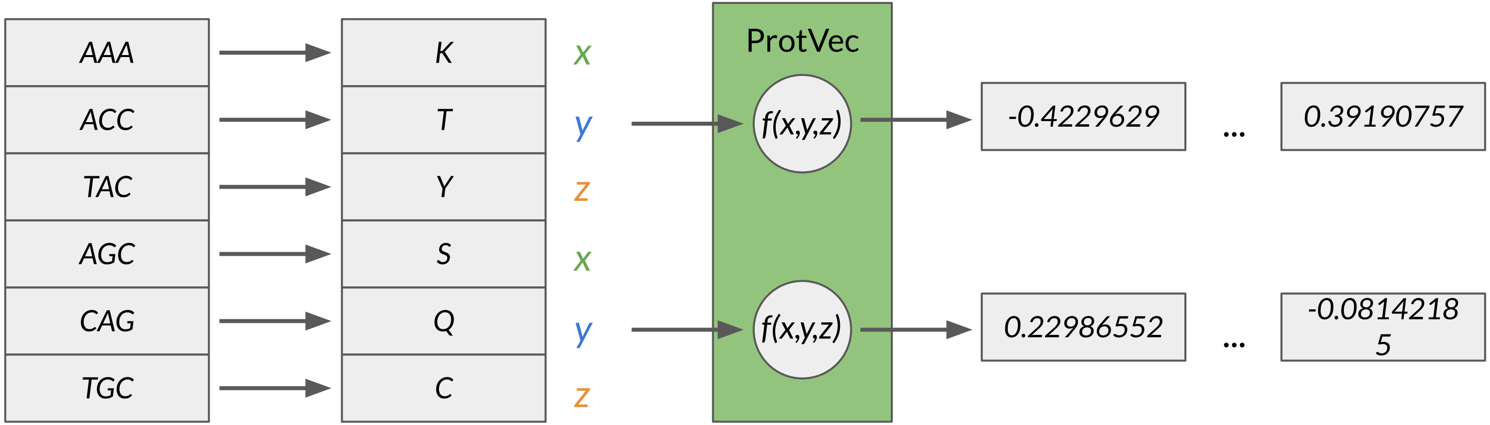


```
[ 3 18  0  0  2]
[ 3 18  0  0  2]
[ 3 18  0  0  2]
[ 3 18  0  0  2]
[ 3 18  0  0  2]
```

ProtVec input

Type 7:

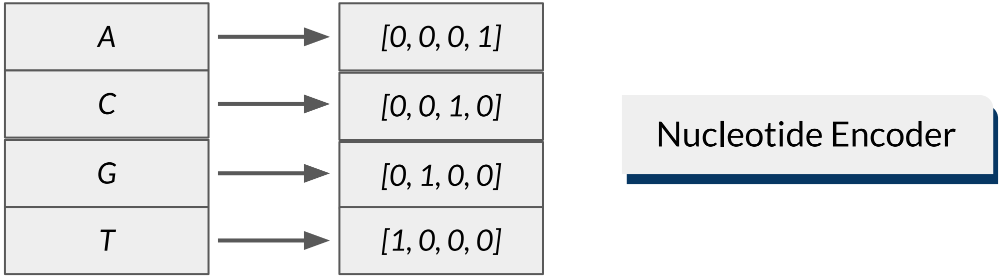
- Word embedding using pre-trained NN
- Input: word of 3 amino acids
- Output: array of 300 real numbers



1.2 ENCODING

Encoding

- Inverted index with frequencies for Frequency input
- Word-embedding for ProtVec input
- One-Hot-Encoding for the rest of input types
    - Encoders specific for each input field (nucleotide, amino acid, event-type , complexity, protein)



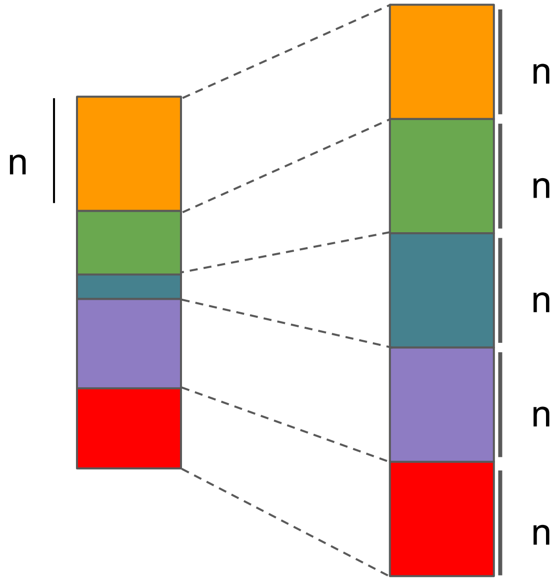
A	→	*[0, 0, 0, 1]*
C	→	*[0, 0, 1, 0]*
G	→	*[0, 1, 0, 0]*
T	→	*[1, 0, 0, 0]*

Nucleotide Encoder

1.3 BALANCING

Balancing (Up-sampling)

- Classes have different number of entries
  - Max is ≈8k
  - Min is ≈2k
- Rebalancing dataset using SMOTE python library on every class that has # entries lower than 8k



3 CLASSIFIERS

Standard approaches

Random Forest

- 100 estimators
- max features set to ‘sqrt’ to provide feature randomness
- Input types:
  - Simple
  - Extended
  - Sequence
  - Sequence extended
  - Sequence Frequency
  - ProtVec

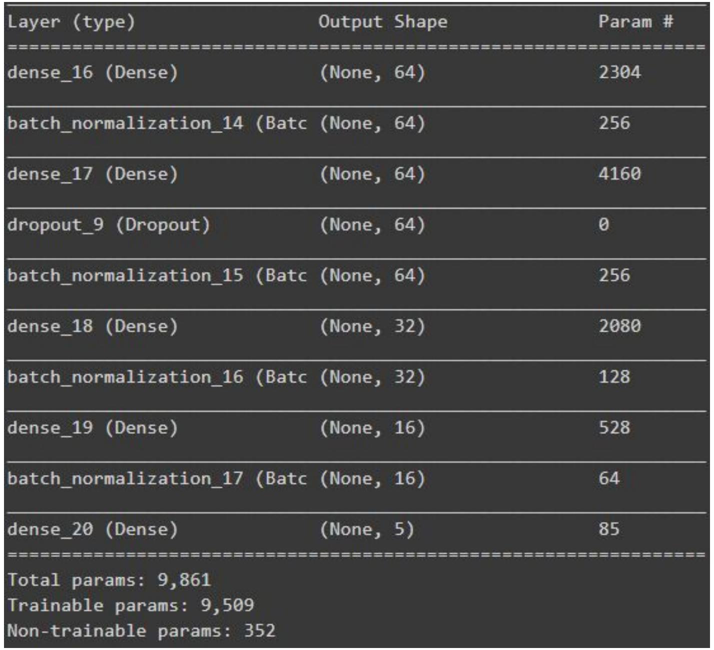
SVM

- A non linear kernel is used, in particular RBF (Radial Basis Function)
- Input types:
  - Simple
  - Extended
  - Sequence
  - Sequence extended
  - Sequence Frequency
  - ProtVec

Neural Network Models

Feed-forward

- 5 fully connected layers with decreasing size and Relu activation function
- loss : categorical cross entropy
  optimizer : adamax with learning rate 0.002
- Custom input types :
  - Extended
  - Sequence
  - Sequence extended
  - Sequence Frequency
  - ProtVec

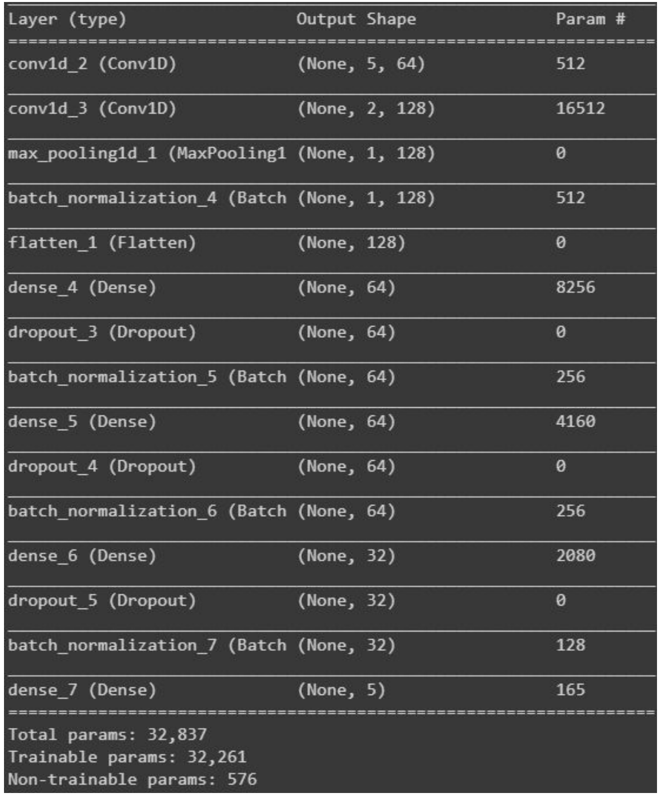


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_16 (Dense)             (None, 64)                2304      
_________________________________________________________________
batch_normalization_14 (Batc (None, 64)                256       
_________________________________________________________________
dense_17 (Dense)             (None, 64)                4160      
_________________________________________________________________
dropout_9 (Dropout)          (None, 64)                0         
_________________________________________________________________
batch_normalization_15 (Batc (None, 64)                256       
_________________________________________________________________
dense_18 (Dense)             (None, 32)                2080      
_________________________________________________________________
batch_normalization_16 (Batc (None, 32)                128       
_________________________________________________________________
dense_19 (Dense)             (None, 16)                528       
_________________________________________________________________
batch_normalization_17 (Batc (None, 16)                64        
_________________________________________________________________
dense_20 (Dense)             (None, 5)                 85        
=================================================================
Total params: 9,861
Trainable params: 9,509
Non-trainable params: 352
```

Neural Network Models - II

CNN

- 2 1D convolutional layers:
  - kernel size 1, activation Sigmoid
  - kernel size 2, activation Relu
- 4 fully connected layers with decreasing size and Relu activation function
- loss : categorical cross entropy
  optimizer : adamax with learning rate 0.002
- Custom input types :
  - Sequence 2D
  - Sequence Frequency 2D

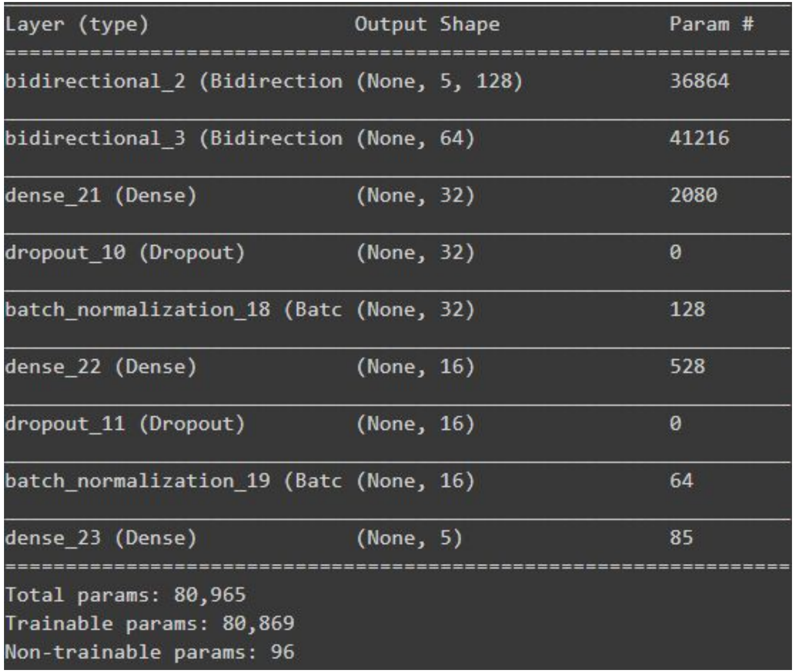


```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_2 (Conv1D)            (None, 5, 64)             512
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 2, 128)            16512
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 1, 128)            0
_________________________________________________________________
batch_normalization_4 (Batch (None, 1, 128)            512
_________________________________________________________________
flatten_1 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
batch_normalization_5 (Batch (None, 64)                256
_________________________________________________________________
dense_5 (Dense)              (None, 64)                4160
_________________________________________________________________
dropout_4 (Dropout)          (None, 64)                0
_________________________________________________________________
batch_normalization_6 (Batch (None, 64)                256
_________________________________________________________________
dense_6 (Dense)              (None, 32)                2080
_________________________________________________________________
dropout_5 (Dropout)          (None, 32)                0
_________________________________________________________________
batch_normalization_7 (Batch (None, 32)                128
_________________________________________________________________
dense_7 (Dense)              (None, 5)                 165
=================================================================
Total params: 32,837
Trainable params: 32,261
Non-trainable params: 576
```

Neural Network Models - III

LSTM

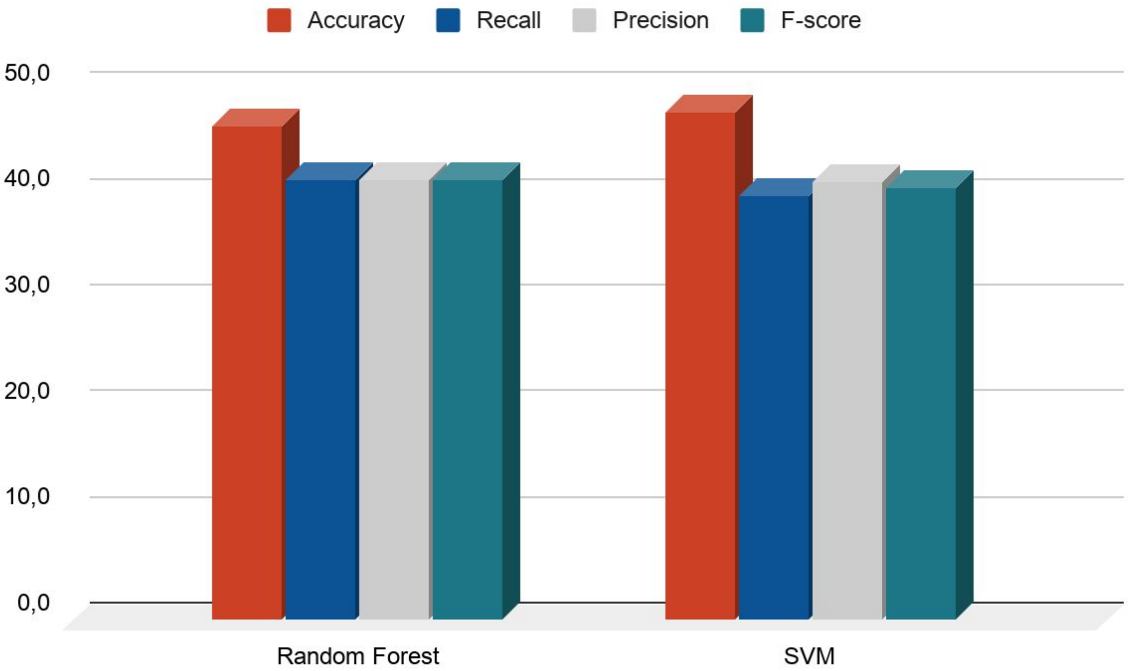
- 2 bidirectional lstm layers with a small recurrent dropout value to avoid fast overfitting
- 3 fully connected layers with decreasing size and Relu activation function
- loss : categorical cross entropy
  optimizer : adamax with learning rate 0.002
- Custom input types :
  - Sequence 2D
  - Sequence Frequency 2D



```
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_2 (Bidirection (None, 5, 128)            36864
_________________________________________________________________
bidirectional_3 (Bidirection (None, 64)                41216
_________________________________________________________________
dense_21 (Dense)             (None, 32)                2080
_________________________________________________________________
dropout_10 (Dropout)         (None, 32)                0
_________________________________________________________________
batch_normalization_18 (Batc (None, 32)                128
_________________________________________________________________
dense_22 (Dense)             (None, 16)                528
_________________________________________________________________
dropout_11 (Dropout)         (None, 16)                0
_________________________________________________________________
batch_normalization_19 (Batc (None, 16)                64
_________________________________________________________________
dense_23 (Dense)             (None, 5)                 85
=================================================================
Total params: 80,965
Trainable params: 80,869
Non-trainable params: 96
```

4 RESULTS

Evaluation - Simple input

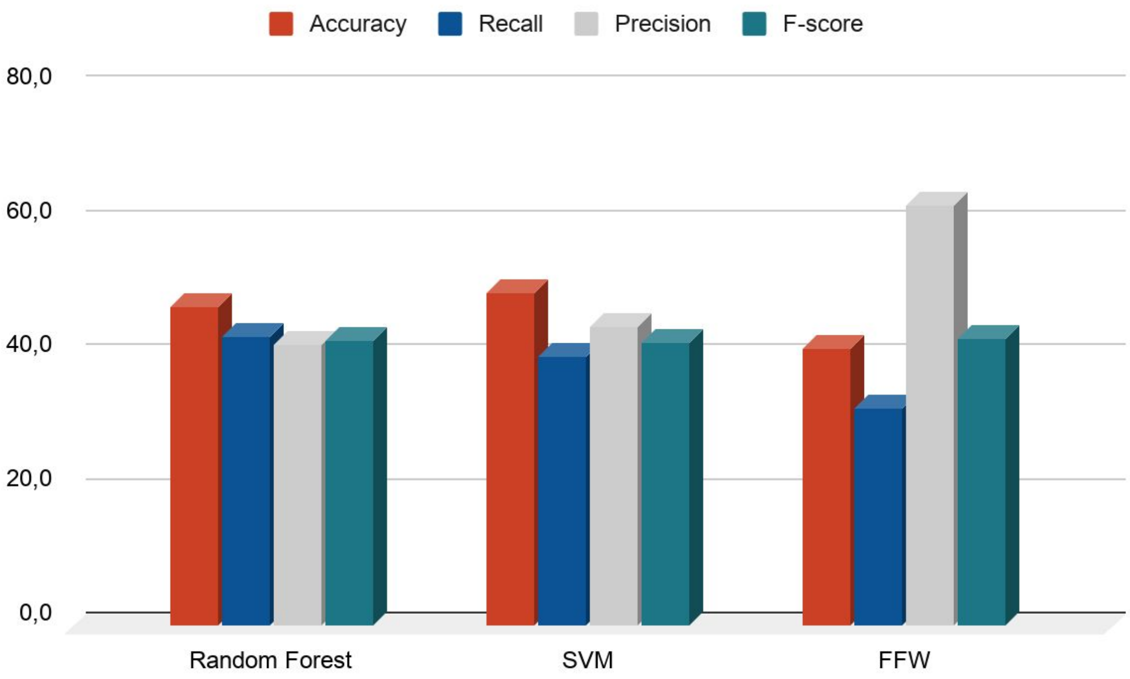


- Due to the very small size of the first and simplest input, it hasn't been tested on the NNs

- The results are not enough satisfying. It follows that the only information regarding the single codon mutation is not representative for our problem

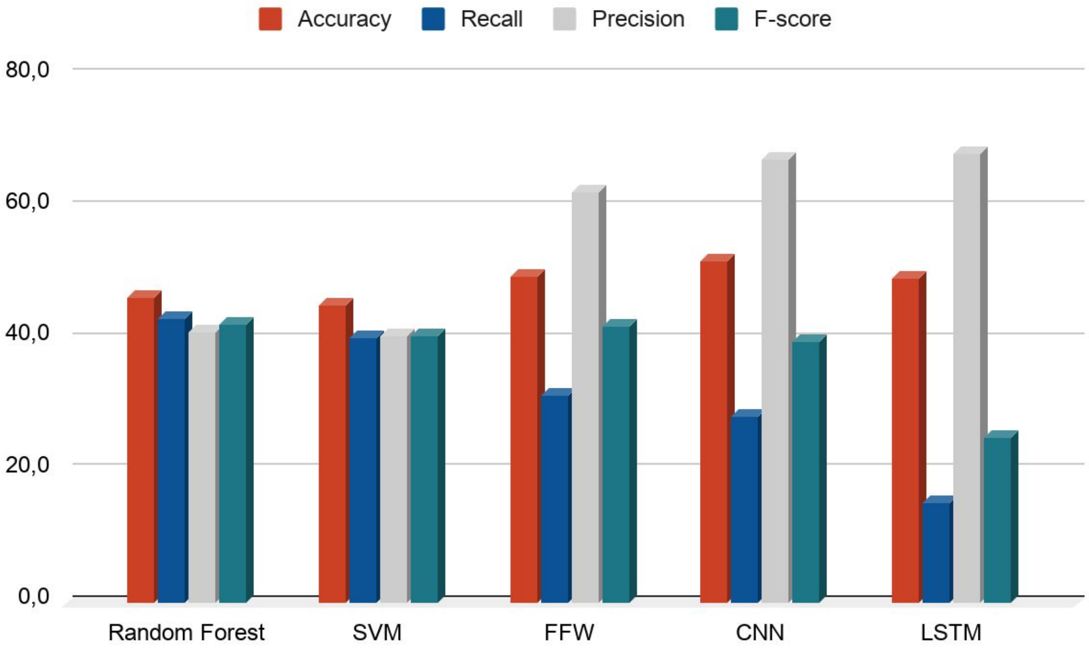
- This leads us to an extension of the input features

Evaluation - Extended input



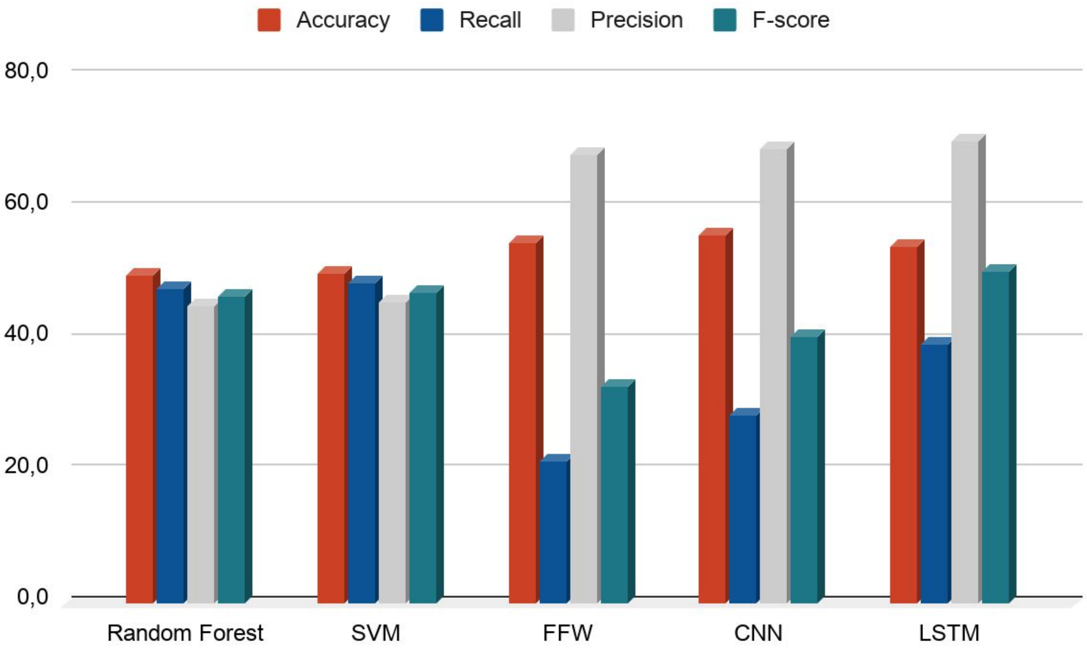
- A slight improvement is accomplished using other features
- The results are still not enough satisfying, in particular for the feed-forward network
- This suggests us to use a different encoding for the input features

Evaluation - Sequence input



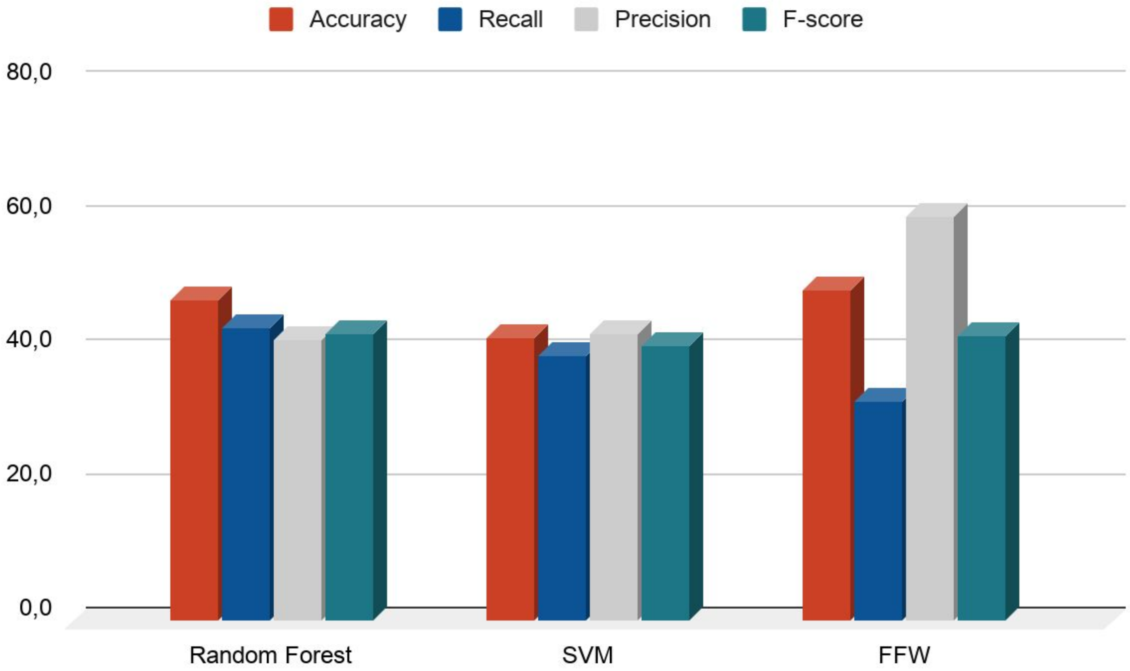
- The introduction of sequences has brought another performance improvement
- It has now sense to use both the CNN and the LSTM on the sequence of the encoded vectors of these subsequences
- Problem: the same sequence might appear in different classes

Evaluation - Frequency input



- Getting the inspiration from text classification problem, the class frequency is now used to encode the subsequences
- We obtain a way smaller and yet better representative input, which translates in better performance in less time
- The CNN and the LSTM achieve the best results

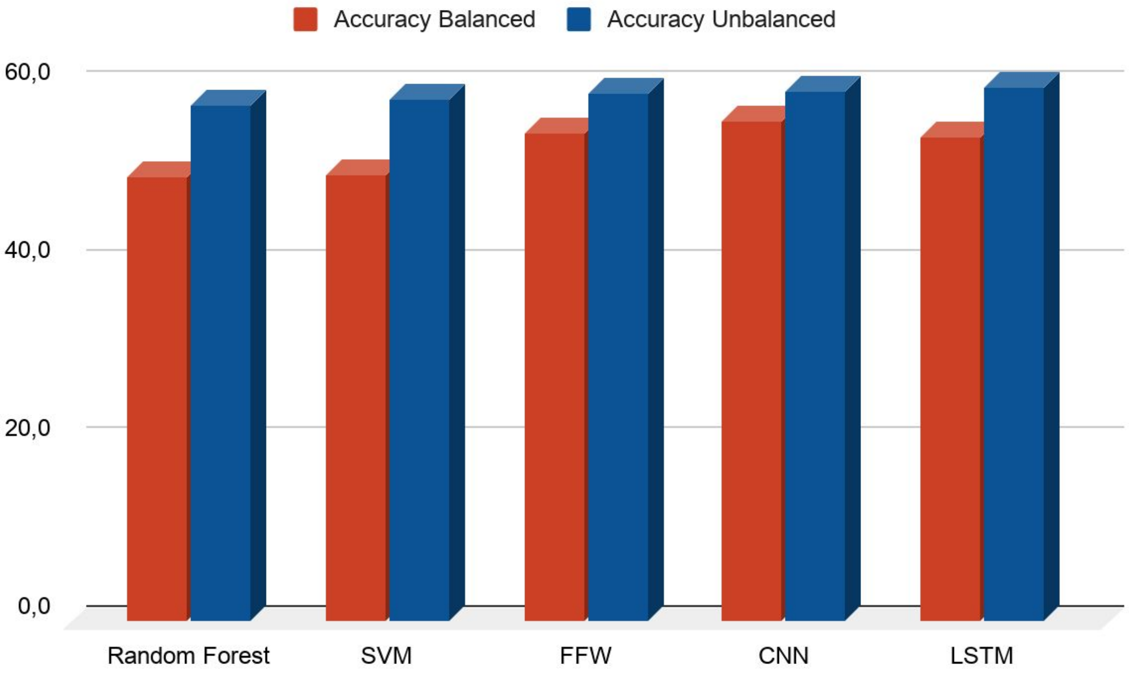
Evaluation - ProtVec input



- Instead of implementing our encoding mechanism for the input features, use pretrained NN for word embedding

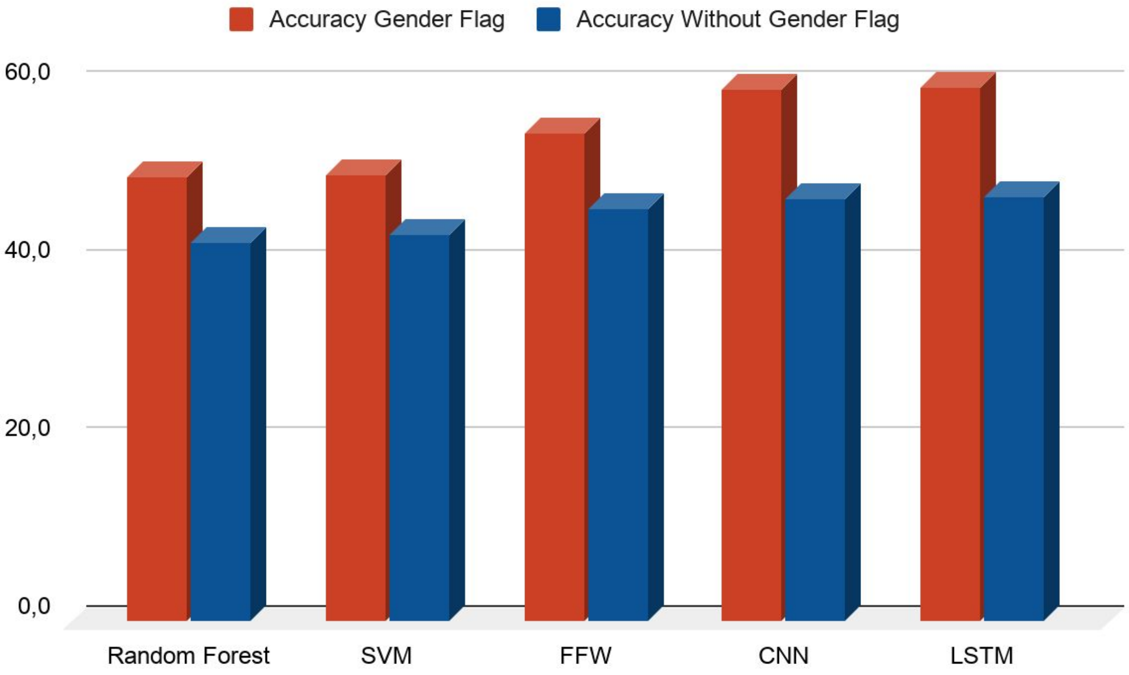
- There's no accuracy increment with respect to one-hot-encoded Sequence input

Evaluation - Balancing



- Another interesting test is the balancing of the dataset
- Only the training set is balanced in up-sampling. The frequency input is used for this test
- The outcome is not what we were expecting. The balancing worsens the performance of every classifier.

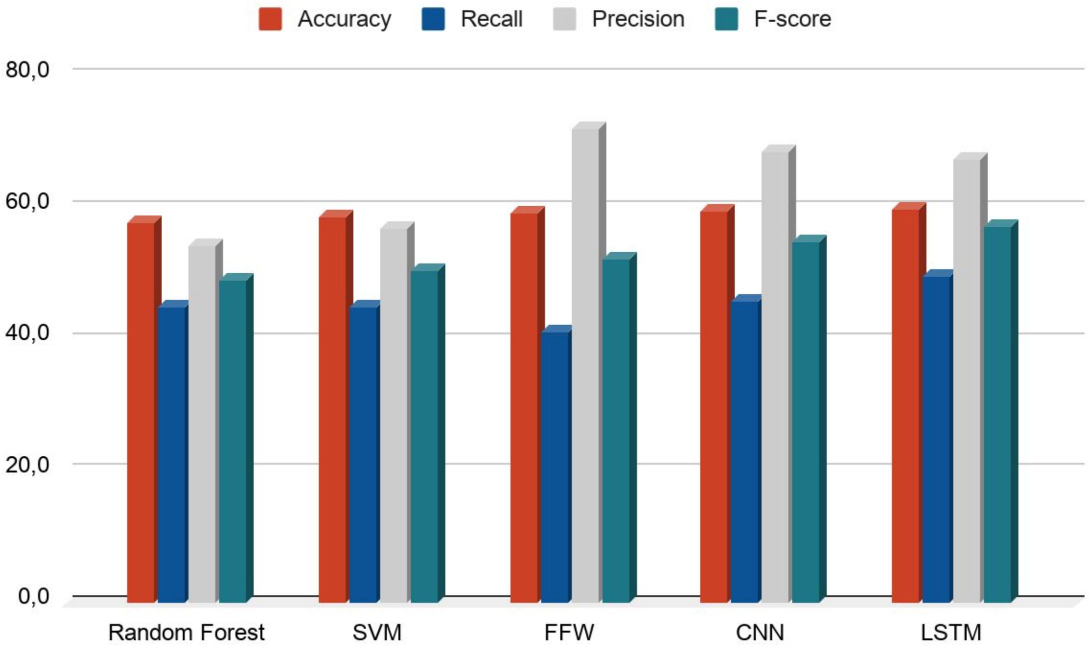
Evaluation - Gender flag



- Some tumors are gender specific, but unfortunately the dataset contains no gender information
- For this test the frequency input has been used
- We have simulated a gender flag removing “Breast Carcinoma” and “Ovarian Carcinoma” classes. It resulted in a sensitive increase of the performance

5 FINAL CONSIDERATIONS

Best Performance



- The best result is obtained using the frequency input, it means that giving a subsequence the right weight improves the class discrimination
- Balancing the dataset worsens the classification
- Just adding a flag on the patient gender boosts significantly the performance

Final insights

Balancing

- The dataset balancing is not working probably because there is no deterministic criteria to perform artificial dataset augmentation, considering only these sequence of codons.
- Synthetic entries, hence, only introduce noise in the dataset

Missing info

- This dataset is not representative for the cancer classification problem.
- Other factors are significative for tumorigenesis, like for example lifestyle and age
- We don’t have the complete set of TP53 gene mutations for each patient, so we had to approximate using the standard gene sequence

References

Original work

Effective Data Mining Technique for Cancer Classification via Mutations in Gene using Neural Network:
https://arxiv.org/ftp/arxiv/papers/1608/1608.02888.pdf

Mutations dataset from:
http://p53.fr/download-the-database

Related works

Continuous Distributed Representation of Biological Sequences for Deep Genomics and Deep Proteomics:
https://arxiv.org/ftp/arxiv/papers/1503/1503.05140.pdf

BioVec Repository:
https://github.com/kyu999/biovec

THANKS FOR YOUR ATTENTION