Deep Learning Approach for Prediction of Advanced Liver Fibrosis in Chronic Hepatitis C Patients

Done By

ARNAOUT, BASSAM; Student ID: 002-25-3288
HAMDAN, AHMAD; Student ID: 002-23-2322

Presented To

Dr. Mohammed Ayoub Alaoui Mhamdi

Dept. of Computer Science
Bishop's University
Sherbrooke, Canada

***Abstract*- Background/Aim**: Using machine learning approaches as non-invasive methods have been used recently as an alternative method in staging chronic liver diseases for avoiding the drawbacks of biopsy. This study aims to evaluate different machine learning techniques in prediction of advanced fibrosis by combining the serum bio-markers and clinical information to develop the classification models. **Methods:** A prospective cohort of 1,385 patients with chronic hepatitis C was divided into two sets – one categorized as mild to moderate fibrosis (F0-F2), and the other categorized as advanced fibrosis (F3-F4) according to METAVIR score.

Keywords—*Liver fibrosis prediction; Machine Learning Algorithm; Hepatitis C virus*

I. INTRODUCTION

In recent years, machine-learning techniques such as classification trees and artificial neural networks (ANN) have been used as prediction, classification, and diagnosis tools. Machine-learning techniques are used in the medical approaches to help using an invasive method in prediction and detection of diseases, such as prediction of fibrosis, cirrhosis, and prediction of response therapy in Hepatitis C Patients [1-3]. We will use the UCI Machine Learning Repository for HCV dataset [4]. We will start with an overview of the common tasks in a machine learning project. A predictive modeling machine learning project can be broken down into main processes:

Data Analysis & Preparation

- Define Problem: Investigate and characterize the problem in order to better understand the goals of the project.
- Analyze Data: Use descriptive statistics and visualization to better understand the data we have available.
- Prepare Data: Use data transforms in order to better expose the structure of the prediction problem to modeling algorithms.

ML Modeling

- Evaluate Algorithms: Design a test harness to evaluate a number of standard algorithms on the data and select the top few to investigate further.
- Improve Results: Use algorithm tuning and ensemble methods to get the most out of well-performing algorithms on the data.
- Present Results: Finalize the model, make predictions and present results.

II. DATA ANALYSIS & PREPARATION

A. Understand the Data

We must understand the data in order to get the best results. The Data has 28 features and an output with 4 classes as:

```
BaselinehistologicalStaging
          1    336
          2    332
          3    355
          4    362
```

```
Age                                int64
Gender                             int64
BMI                                int64
Fever                              int64
Nausea-Vomting                     int64
Headache                           int64
Diarrhea                           int64
Fatigue-generalized-bone-ache      int64
Jaundice                           int64
Epigastric-pain                    int64
WBC                                int64
RBC                                int64
HGB                                int64
Plat                               int64
AST1                               int64
ALT1                               int64
ALT4                               int64
ALT12                              int64
ALT24                              int64
ALT36                              int64
ALT48                              int64
ALTafter24w                        int64
RNABase                            int64
RNA4                               int64
RNA12                              int64
RNAEOT                             int64
RNAEF                              int64
Baseline-histological-Grading      int64
BaselinehistologicalStaging        int64
dtype: object
```

B. Descriptive Statistics

We now have a better feeling for how different the attributes are. The min and max values as well as the means vary a lot. We are likely going to get better results by rescaling the data in some way

	Age	Gender	BMI	Fever	Nausea-Vomting	Headache	Diarrhea	Fatig generaliz bone-a
count	1385.000000	1385.000000	1385.000000	1385.000000	1385.000000	1385.000000	1385.000000	1385.000
mean	46.319134	1.489531	28.608664	1.515523	1.502527	1.496029	1.502527	1.498
std	8.781506	0.500071	4.076215	0.499939	0.500174	0.500165	0.500174	0.500
min	32.000000	1.000000	22.000000	1.000000	1.000000	1.000000	1.000000	1.000
25%	39.000000	1.000000	25.000000	1.000000	1.000000	1.000000	1.000000	1.000
50%	46.000000	1.000000	29.000000	2.000000	2.000000	1.000000	2.000000	1.000
75%	54.000000	2.000000	32.000000	2.000000	2.000000	2.000000	2.000000	2.000
max	61.000000	2.000000	35.000000	2.000000	2.000000	2.000000	2.000000	2.000

C. Understand the Data With Visualization

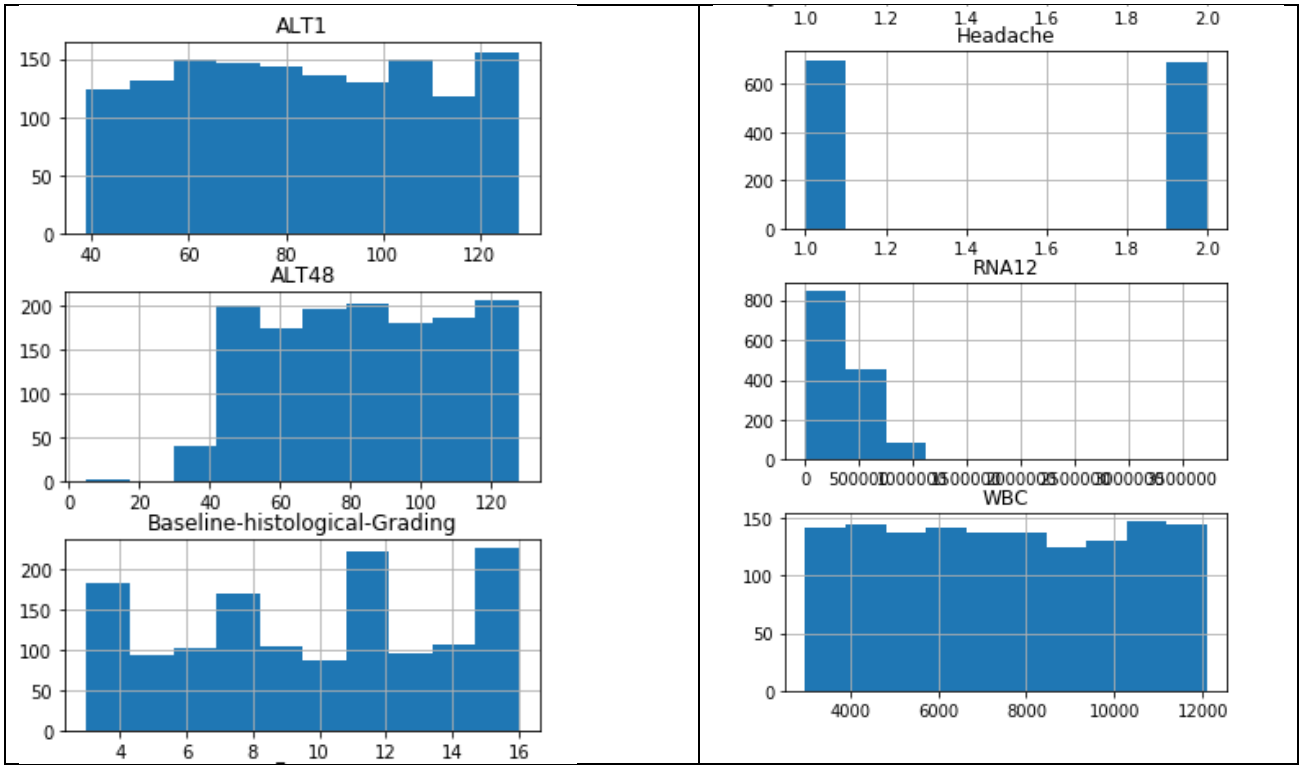
Below is Univariate Histograms for some of the attributes.

We can see that no attributes have an exponential distribution. We can also see that perhaps some attributes have a Gaussian or nearly Gaussian distribution. This is interesting because many machine learning techniques assume a Gaussian univariate distribution on the input variables. Such as “ALT1”

This is useful to note as we can use algorithms that can exploit this assumption (Gaussian distribution).

Some has binomial distribution such as “Headcahe” attribute.

It also looks like some attributes may be skewed Gaussian distributions, which might be helpful later with transforms such as "RNA12" attribute.



D. Discretization-Criteria

Since the outcome class has four output as 1,2,3,4. We will divide into two sets – one categorized as mild to moderate fibrosis (F0-F2) being as 0, and the other categorized as advanced fibrosis (F3-F4) being as 1

```
BaselinehistologicalStaging
0    668
1    717
dtype: int64
```

E. Log transform certain features (skewed distribution)

We will apply log transformation to those attribute who has a skewed distribution. The attributes are: ['RNA12', 'RNAEOT', 'RNAEF']

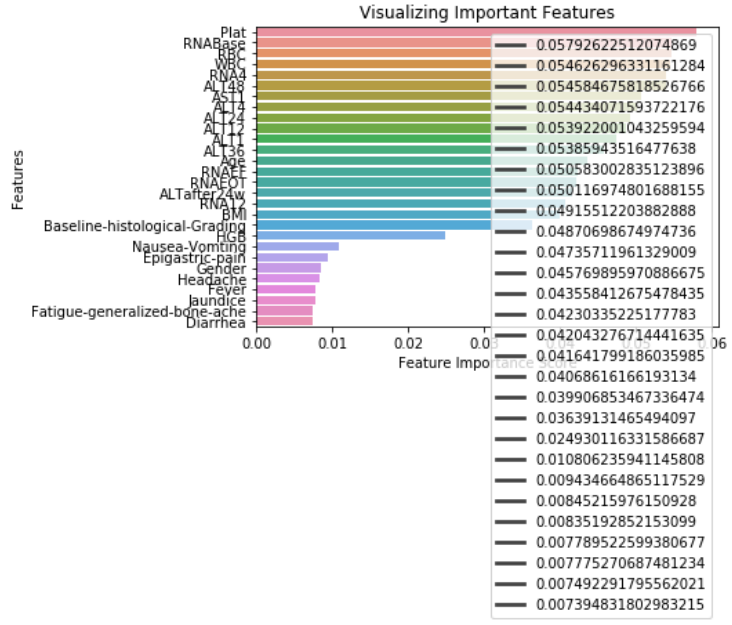
After log transformation is applied, we notice that now we do not have any attribute with skew distribution as show below.

```
Age                                0.006835
Gender                             0.041932
BMI                               -0.035573
Fever                             -0.062191
Nausea-Vomting                    -0.010119
Headache                           0.015902
Diarrhea                          -0.010119
Fatigue-generalized-bone-ache      0.004337
Jaundice                          -0.004337
Epigastric-pain                   -0.015902
WBC                                0.016081
RBC                               -0.047663
HGB                               -0.046887
Plat                               0.032442
AST1                               0.030132
ALT1                               0.017926
ALT4                               0.023289
ALT12                             -0.011167
ALT24                             -0.033095
ALT36                             -0.041431
ALT48                             -0.044457
ALTafter24w                       -0.111222
RNABase                            0.018666
RNA4                               0.004409
RNA12                             -0.889942
RNAEOT                            -0.913121
RNAEF                             -0.920676
Baseline-histological-Grading     -0.105961
BaselinehistologicalStaging       -0.070879
dtype: float64
```

F. Finding Important Features Using RandomForestClassifier

We have created a random forests model & used the feature importance variable to see feature importance scores. Then we visualized these scores as below:

```
Finding Important Features
```



We can see that the last 8 attributes have low score compared to others. We can eliminate them for better training performance.

G. Standardize Data

When the data is comprised of attributes with varying scales, many machine learning algorithms can benefit from rescaling the attributes to all have the same scale. Often this is referred to as normalization and attributes are often rescaled into the range between 0 and 1. This is useful for optimization algorithms used in the core of machine learning algorithms like gradient descent. It is also useful for algorithms that weight inputs like regression and neural networks and algorithms that use distance measures like k-Nearest Neighbors. We can rescale the data using scikit-learn using the MinMaxScaler class.

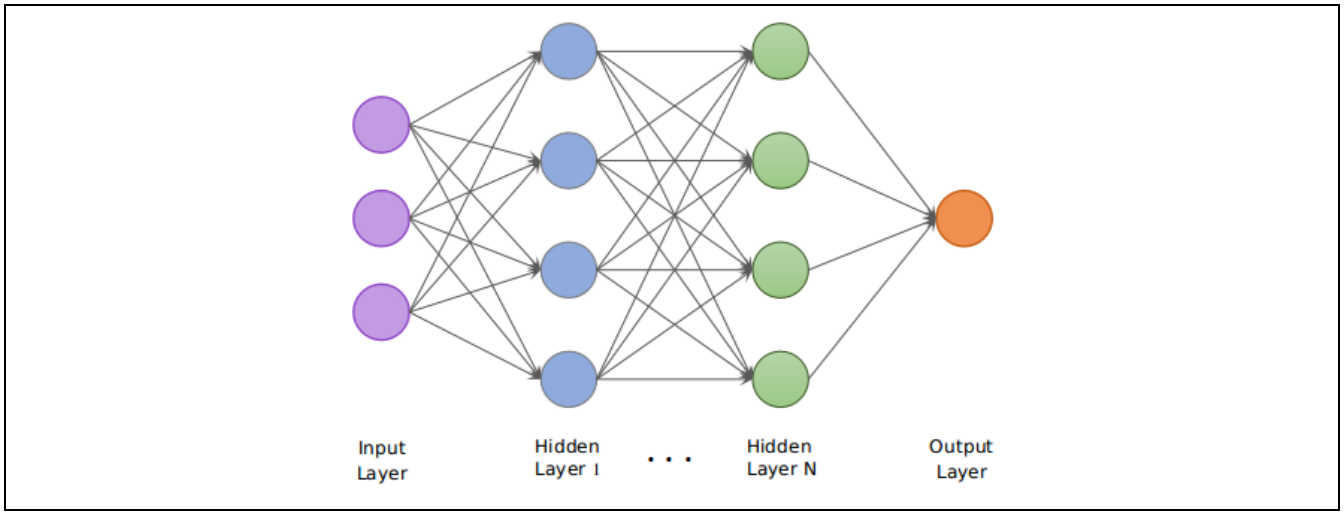
RBC	HGB	Plat	AST1	ALT1	ALT4	ALT12	ALT24	ALT36	ALT48	ALTaft
713	0.8	0.143266	0.674157	0.505618	0.146067	0.786517	0.471910	0.000000	0.000000	
974	0.0	0.272415	0.584270	0.943820	0.629213	0.404494	0.831461	0.422764	0.959350	
509	0.4	0.438431	0.831461	0.112360	0.629213	0.764045	0.865169	0.000000	0.000000	
798	0.0	0.400477	0.044944	0.280899	0.786517	0.460674	0.550562	0.349593	0.585366	
183	0.2	0.709406	0.674157	0.730337	0.314607	0.101124	0.910112	0.723577	0.691057	
...	...	...	...	...	...	...	...	...	...	
022	1.0	0.820578	0.932584	0.224719	0.438202	0.752809	0.988764	0.471545	0.317073	
620	0.0	0.170572	1.000000	0.707865	0.292135	0.674157	0.775281	0.747967	0.479675	
250	0.8	0.264824	0.247191	0.606742	0.943820	0.247191	0.865169	0.666667	0.276423	
041	0.0	0.845966	0.348315	0.651685	0.730337	0.393258	0.089888	0.349593	0.617886	
954	0.8	0.326727	0.134831	0.977528	0.000000	0.325843	0.853933	0.479675	0.536585	

H. Create a Validation & Test Dataset

We will split the loaded dataset into two, 67% of which we will use to train our models and 33% that we will hold back as a validation & test dataset.

```
Training set has 927 samples.
Validation set has 412 samples.
Testing set has 46 samples.
```

III. DEEP LEARNIG MODELING (FULLY CONNECTED NEURAL NETWORK)



A. Setting up the Architecture

The first thing we have to do is to set up the architecture. Let's first think about what kind of neural network architecture we want.

Since the data is too small, we want to have these layers:

- Input Layer: 28 features
- Hidden layer 1: 12 neurons, ReLU activation
- Output Layer: 1 neuron, Sigmoid activation

B. Filling in the best numbers

Now that we've got our architecture specified, we need to find the best numbers for it. Before we start our training, we have to configure the model by:

- Telling it which algorithm we want to use to do the optimization
- Telling it what loss function to use
- Telling it what other metrics we want to track apart from the loss function

We will use stochastic gradient descent optimizer for the compiler & the loss function for outputs that take the values 1 or 0 is called binary cross entropy.

Before training a model, we need to configure the learning process, which is done via the compile method. It receives three arguments:

- An optimizer. This could be the string identifier of an existing optimizer (such as sgf, rmsprop or adagrad).
- A loss function. This is the objective that the model will try to minimize. It can be the string identifier of an existing loss function (such as categorical_crossentropy or mse).
- A list of metrics. For any classification problem we will want to set this to metrics=['accuracy']. A metric could be the string identifier of an existing metric or a custom metric function.

we're done with specifying our architecture! To see a summary of the full architecture:

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_13 (Dense)             (None, 12)                348
_________________________________________________________________
dense_14 (Dense)             (None, 1)                 13
=================================================================
Total params: 361
Trainable params: 361
Non-trainable params: 0
_________________________________________________________________
```

C. Training the model

We can now see that the model is training! By looking at the numbers, we should be able to see the loss decrease and the accuracy increase over time.

```
927/927 [==============================] - 0s 106us/step - loss: 0.6279 - acc: 0.6494 - val_loss: 0.7274 - val_acc: 0.5146
Epoch 97/100
927/927 [==============================] - 0s 108us/step - loss: 0.6270 - acc: 0.6419 - val_loss: 0.7214 - val_acc: 0.5049
Epoch 98/100
927/927 [==============================] - 0s 108us/step - loss: 0.6266 - acc: 0.6516 - val_loss: 0.7282 - val_acc: 0.5291
Epoch 99/100
927/927 [==============================] - 0s 105us/step - loss: 0.6276 - acc: 0.6321 - val_loss: 0.7302 - val_acc: 0.5316
Epoch 100/100
927/927 [==============================] - 0s 105us/step - loss: 0.6268 - acc: 0.6483 - val_loss: 0.7375 - val_acc: 0.5049
```

Evaluating our data on the test set:

```
1 model.evaluate(X_test, Y_test)[1]
```

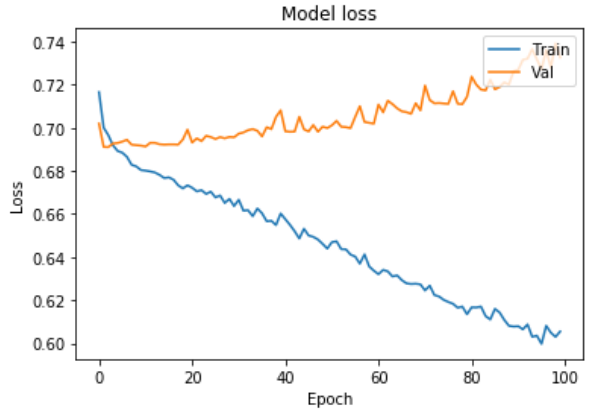
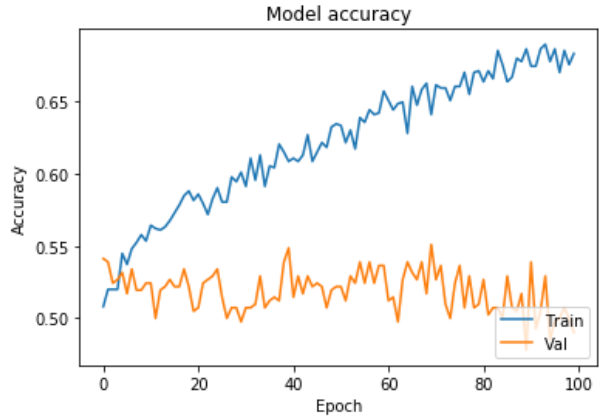
```
46/46 [==============================] - 0s 230us/step
0.3695652212785638
```

```
46/46 [==============================] - 0s 108us/step
0.4782608734524768
```

	precision	recall	f1-score	support
0.0	0.35	0.36	0.36	22
1.0	0.39	0.38	0.38	24
accuracy			0.37	46
macro avg	0.37	0.37	0.37	46
weighted avg	0.37	0.37	0.37	46

D. Visualizing Loss and Accuracy

How do we know if our model is currently over fitting?

We want to visualize the training loss and the validation loss	We can also visualize the training accuracy and the validation accuracy
	

Since the training set does not match up with improvements to the validation set, it seems like over fitting is a huge problem in our model.

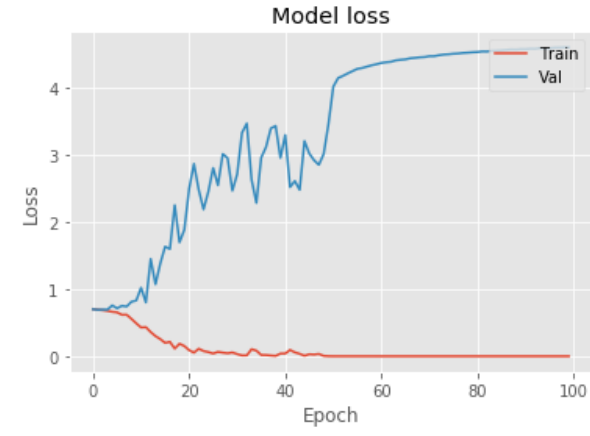
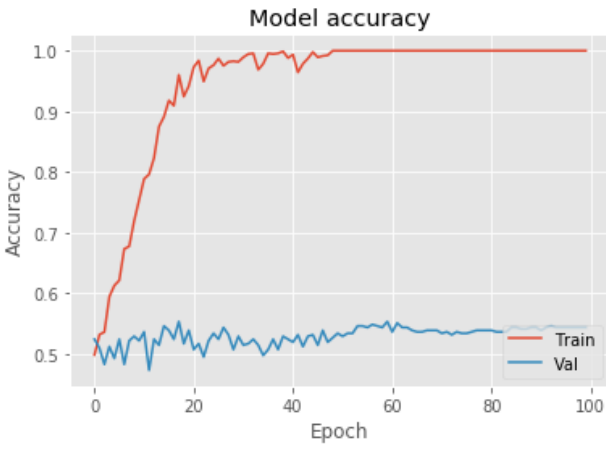
E. Adding Regularization to our Neural Network

For the sake of introducing regularization to our neural network, let's formulate with a neural network that will badly overfit on our training set. We'll call this Model 2.

Here, we've made a much larger model and we've use the Adam optimizer. Adam is one of the most common optimizers we use, which adds some tweaks to stochastic gradient descent such that it reaches the lower loss function faster.

```
1 model_2.evaluate(X_test, Y_test)[1]
```

```
46/46 [==============================] - 0s 156us/step
0.413043482148129
```

We want to visualize the training loss and the validation loss	We can also visualize the training accuracy and the validation accuracy
	

This is a clear sign of over-fitting. The training loss is decreasing, but the validation loss is way above the training loss and increasing (past the inflection point of Epoch 20)

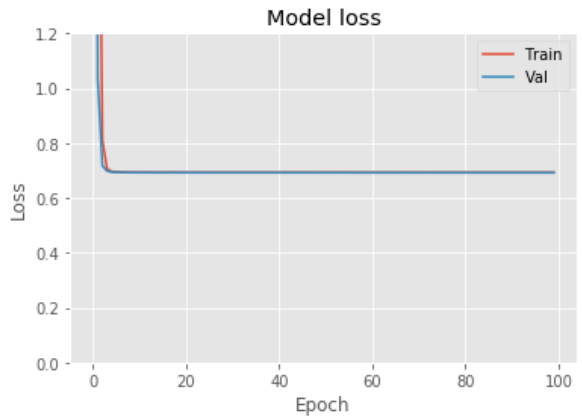
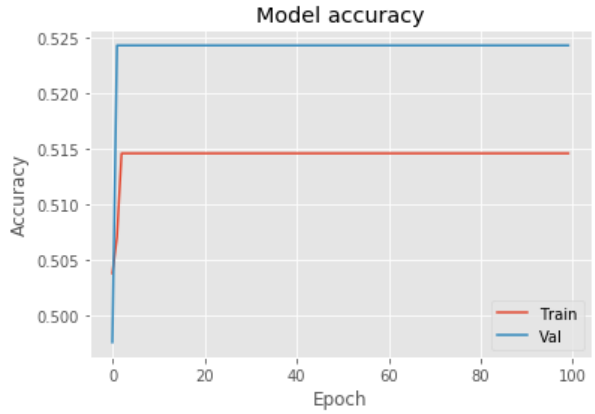
We can see a clearer divergence between train and validation accuracy as well

Now, let's try out some of our strategies to reduce over-fitting (apart from changing our architecture back to our first model). we'll incorporate L2 regularization and dropout here.

There are two main differences between Model 3 and Model 2:
Difference 1: To add L2 regularization
Difference 2: To add Dropout (0.3)

```
1 model_3.evaluate(X_test, Y_test)[1]
```

```
46/46 [==============================] - 0s 199us/step
0.521739134322042
```

We want to visualize the training loss and the validation loss	We can also visualize the training accuracy and the validation accuracy
	

We can see that the validation loss much more closely matches our training loss.
Compared to our model in Model 2, we've reduced overfitting substantially! And that's how we apply our regularization techniques to reduce overfitting to the training set.

F. Hyper Parameters Tuning

Hyper Parameters	**Result**	**Best Accuracy Rate**
Tune Batch Size and Number of Epochs	{'batch_size': 80, 'epochs': 100}	Best: 0.530744
Tune the Training Optimization Algorithm	{'optimizer': 'SGD'}	Best: 0.519957
Tune Learning Rate and Momentum	{'learn_rate': 0.2, 'momentum': 0.8}	Best: 0.541532
Tune Network Weight Initialization	{'init_mode': 'lecun_uniform'}	Best: 0.527508
Tune the Neuron Activation Function	{'activation': 'softsign'}	Best: 0.532902
Tune Dropout Regularization	{'dropout_rate': 0.8, 'weight_constraint': 2}	Best: 0.536138
Tune the Number of Neurons in the Hidden Layer	**{'neurons': 25}**	**Best: 0.543689**

G. Comparison between ML & DL Approach.

When we apply the ML algorithm on the same data set with same filtering process, we get the best accuracy rate among ML algorithms which was SVM with accuracy rate of

SVM Accuracy Rate	Fully Connected Neural Network Accuracy Rate
Prediction on Test data: [[10 12] [15 9]] precision recall f1-score support 0.0 0.40 0.45 0.43 22 1.0 0.43 0.38 0.40 24 accuracy 0.41 46 macro avg 0.41 0.41 0.41 46 weighted avg 0.41 0.41 0.41 46 Accuracy of SVM Model : 41.30434782608695 %	**Best: 0.543689**

We can conclude the using CNN on Modeling such DataSet is the best options for future prediction.

REFERENCES

[1] Laurent Castera. Noninvasive methods to assess liver disease in patients with hepatitis B or C. GASTROENTEROLOGY. 2012; 142:1293–1302.

[2] Li Zhang, Qiao-ying LI, Yun-you Duan, Guo-zhen Yan, Yi-lin Yang and Rui-jing Yang. Artificial neural network aided non-invasive grading evaluation of hepatic fibrosis by duplex ultrasonography. BMC Medical Informatics and Decision Making. 2012; 12:55.

[3] Mahmoud ElHefnawi, Mahmoud Abdalla, Safaa Ahmed, Wafaa Elakel, Gamal Esmat, Maissa Elraziky, Shaima Khamis, et al. Accurate prediction of response to Interferon-based therapy in Egyptian patients with Chronic Hepatitis C using machine-learning approaches. IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). 2012; 771-778

[4] https://archive.ics.uci.edu/ml/datasets/Hepatitis+C+Virus+%28HCV%29+for+Egyptian+patients