JOURNAL OF EMERGING INVESTIGATORS

Machine Learning and Data Shapley Data Selection to Predict Cirrhosis Stages

David Jose Florez Rodriguez[1]
[1] Electrical Engineering, Stanford University, Palo Alto, California

Student Authors (high school)
Ethan Daniel Taylor, Joseph Chai*, Joyce Zheng*, Juliana Maria Gaitan*, Paulos Waiyaki*, Tara Maria Salli**

*These authors contributed equally to this work,

Summary

Cirrhosis is a common and deadly disease that requires the time and experience of a doctor to diagnose. We hypothesize that machine learning empowered by the Data Shapley data selection algorithm could diagnose the stage of cirrhosis to a higher validation accuracy than a machine learning model without such data selection in its training. This could help save doctors time and hospitals and patients money. We sourced the dataset from Kaggle and the data was collected by Mayo Clinic. We first had to design preprocessing steps to fit an AI to the data and define a method for handling NaNs in the data. Then we explored potential models, settling on a TensorFlow neural network classifier. Research explored the hyperparameter space to obtain an ideal neural network for our task. The Data Shapley algorithm then ran with the best previously defined model to test whether data selection could improve accuracy when using such a small dataset. Data selection had some merit in selecting better training sets than random selection of a fixed-sized subset of the dataset, but training on the whole training set gave the best results. If thoroughly explored, the method could accurately predict the stage. It could become an invaluable tool for the healthcare industry.

Introduction

Cirrhosis is among the most common causes of death worldwide. One in every 400 adults in the US have cirrhosis and in 2017, cirrhosis was ranked as the 11th leading cause of death. With no definitive cure, early detection is paramount for a patient's survival. Cirrhosis affects a patient's liver, gradually replacing healthy liver cells with scarred cells. Due to its progressive nature, cirrhosis can take several years to fully develop. This makes it extremely difficult to detect early on, significantly decreasing a patient's chances of survival.

There are four stages to Cirrhosis [1]. The first stage begins with inflammation of the bile duct and/or liver; in the second stage, the inflammation transforms into scar tissue; the third stage involves the liver losing its ability to function optimally due to the scarring; and finally, the fourth stage results in liver failure and a high risk for developing liver cancer.

Cirrhosis can be detected either by radiology testing or a needle biopsy of the liver. Both methods are costly and can become a significant financial burden for the patient and their family; therefore, coming up with less invasive and cost-effective ways to triage a patient's risk for cirrhosis with common medical knowledge could increase the chances of early detection.

With this goal in mind, this work tests various AI models to predict a patient's stage of Cirrhosis purely based on accessible medical data.

Results

The initial set of experiments utilized the website superbio.ai [2] to conduct tests; however, NaN values[1] within the data obstructed testing. To clean the dataset, two transformations of the data explored NaN handling in numerical features: 'MedNaN' – where NaN values were replaced with medians – and 'ZeroNan' – where NaN values were made zeros. After defining these two datasets, various models explored architectures, learning rates (LR), whether the predicted stage variable was categorical or numerical, and different activation functions: Relu, Tanh, and Sigmoid. In total, experiments explored a total of 180 models.

As shown in figure 1, the best model from those experiments was a MedNan, categorical, Relu model with four layers, 64 cells per layer, and a 0.04 learning rate – this model had an accuracy rate of 57%. The worst model was also a MedNan categorical Relu model with three layers and a 0.1 learning rate with 18% accuracy.

As shown in figure 2 the best MedNan results performed better than the all results from ZeroNan. Among ZeroNan tests, a categorical, Sigmoid model with LR 0.0003 and 22% accuracy was the worst, and a categorical, Relu model with 0.1 LR and 51% accuracy was the best. In comparison, the best and worst results from MedNan were, respectively, a categorical, Relu model with 0.04 LR and 57% accuracy, and a categorical, l Sigmoid model with 1 LR and 5% accuracy. Additionally, regression architectures typically performed poorly overall as they gave nonsensical numbers for the stages, including negatives and values above four. When it came to activation functions as shown in figures 1 and 2, Relu tended to have higher accuracy rates than Sigmoid and Tanh, specifically for MedNan results. The best Tanh model was a categorical Tanh with a 0.003 LR, three layers, and 55% accuracy. The best Sigmoid model, on the other hand, was a categorical Sigmoid with a 0.04 LR, three layers, and 47% accuracy.

[1] Not a Number, a datatype in python and general data science to denote missing information.

With a clear preference for types of models and optimal handling of NaN values, the next experiments explored more architectures[2], hyperparameters[3], L2 regularization coefficients (REG), and learning rates All of these experiments ran on python. Training included 50 epochs for each model. Of all the models run, accuracies were in the range of 20% to 48%. As shown in figure _ the best (three) models each had 48.28% accuracy and their details are below:

1. LR (0.003), REG (0.03), and cell numbers (100,100,100)
2. LR (0.003), REG (0.03), and cell numbers (60,50,40,30,20)
3. LR (0.003), REG (0.03), and cell numbers (50,100,50,20)

The worst models had 21.84% accuracy rates, with the following details:

4. LR (0.003), REG (0.01), and cell numbers (100,100,100)
5. LR (0.0003), REG (0.03), and cell numbers (60,50,40,30,20)
6. LR (0.003), REG (0.01), and cell numbers (200,200)

In the final stage of experiments, a data selection algorithm gave every patient a ‘value’, approximating how much it would help in training a model. The optimal architecture from previous experiments was then trained on the top 50, 100, 150, 200, and 250 patients according to data selection. We compared these 5 models and their performances to models with the same optimal architecture but trained on random sets of 50, 100, 150, 200, and 250 patients, and the subset of patients included only patients with no missing numerical data.

Data Selection’s validation accuracy peaked at 200 patients at about 45.9%. Random selection had the highest accuracy rate from this stage of experiments between the two data sets on the 250 patients subset of the data; nevertheless, the optimal architecture from the previous stage had the best accuracy overall when training on the whole training set.

Data selection outperformed random selection for most models ran, save for the models with 250 patients, where data selection had 41.4% accuracy and random selection had 47.1% accuracy.

Discussion

[2] Architectures herein are described such that a model with 3 layers, where the first has 100 cells and the next two have 40 cells, would be denoted (100,40,40). The input data (size 32) and output (size 4) are excluded in these architecture shorthand.

[3] In supervised ML, parameters that can’t be optimized with gradient descent, and thus must be varied and tested iteratively to find an optimal choice.

The original experiment constructed a model by attempting to use a linear regression model to predict the stage. However, this initial experiment failed, and the model obtained poor results: the predicted stage was often outside the desired range between one and four (see figure 4). Although stages are numbered and progressive, their numbering is more symbolic of their being sequential than representative of their relative value. The trial failed, likely because linear regression assumes the stage prediction is a linear function of the input variables. For this dataset, which contained 32 columns in the input representing 18 numerical and categorical variables (where categorical variables were made one-hot vectors, increasing the column count), results suggest the stage prediction task is non-linear.

The following experiment on the superbio.ai site included neural network [3] regressors and classifiers. The experiment trained on two versions of the data, MedNan and ZeroNan, tested each with various hyperparameters, and predicted stage as either a categorical or numerical feature type. The 180 different tests yielded three key findings.

1. MedNan results performed better than ZeroNan results. This might be because replacing NaN values with the median values of that column further reinforces the existing distribution for a given feature and makes the resulting feature more consistent. Replacing the Nan values with zeros reduces the accuracy, possibly due to skewing the distribution for that feature by introducing outlier values (e.g.: setting an unknown age to 0 makes little sense).

2. Regression architecture results were insufficient when compared to results from categorical models. This might be due to the stages being categorical variables by their medical definition despite their clear progressive nature, so running a Numerical test on it is ineffective. In addition, a lot of the variables were categorical (drug administered, sex, and stage), and their presence in the input may favor a similar categorical structure in the output. The predicted stages had the same issues found when using a linear regressor.

3. The Relu activation function was significantly more accurate than the rest of the models.

Focusing solely on MedNan categorical results, lower learning rates and deeper models tended to perform better in experiments. This may be due to deeper models having more parameters

and thus a greater capacity to learn. With adequate architectures and NaN handling methods that produced the best results (MedNan, Categorical, and Relu), the research progressed to later stages.

The next step in the research was training hyperparameters and optimizing the architecture even further (various amounts of layers and cells per layer). This step integrated a Data Shapley [4] algorithm to the existing models implemented in Python3 through TensorFlow [5]. This data selection algorithm employing said model gave values to each data point. Chosen 'highly valuable' subsets of the data trained multiple models with this optimal architecture, while other models trained on random subsets of the data, and one model trained on a subset including only patients with no missing data. The resulting performances validated data selection for creating subsets over randomly created subsets four out of five times when making a training subset. For all other comparisons, there was an average of 11.1% difference between the results from data selection and the results from random selection (max being 15.9% with 50 patients, and minimum being 8% with 200 patients). Models with increasing amounts of selected training data had no observable pattern. Random selection, on the other hand, improved in accuracy when the training set grew from 50 to 250. The models trained on selected data also outperformed models trained on the subset with no filled-in numerical NaNs (276 patients in total, with an accuracy of 41.4%). This is possibly due to the limited data size, which could make it more valuable to have more patients with mostly complete data and filled in NaNs than to have perfect data, but fewer patients. Ultimately, the model trained on all the data (325 patients and an accuracy of 48.3%) had the highest performance.

In conclusion, the hypothesis of data selection boosting a machine learning model's performance was supported at smaller dataset sizes (*n* less than 200) but not at the scale of the complete dataset. Data selection's improvement in selecting training sets when compared to random subsets, however, suggests that data selection may be useful in some cases.

Methods and Materials

Codebase

All code supporting this work in the preprocessing, architecture and hyperparameter search, and the data selection stages of the research is available at https://github.com/elsirdavid/Shtem2022. The tests using superbio.ai required no code and are possible to repeat given our public preprocessing code and the public dataset from Kaggle.

Data Preprocessing

The dataset used for training and validation was from Kaggle [6] [7]. The Mayo Clinic [8] gathered the data from 418 cirrhosis patients, each with 20 data points. The data contained some issues, however. Six patients were missing a stage; so they were removed from the dataset, as the stage, is an essential part of this research. Some patients' data contained 'not a number' (NaN) values for numerical data points, and several categorical values existed that needed to be converted to one-hot encoding for the neural network. The encoding converted all the potential values in the category into their own separate categories. These then had binary values to represent if the category was present. For example, in the category "Drug Administered" there can be many categories, such as 'placebo', 'D-penicillamine', or 'N/A'. One-hot encoding makes a new column in the data for each category (placebo, D-penicillamine, ...) and uses 1s and 0s to denote if this patient belongs to said category. These purely numerical values can then train a machine learning [9] model. Numerical NaNs differ from categorical NaNs in this way. When a categorical data point has a NaN, the NaN value can simply be another category within the feature. However, numerical NaNs need to be filled in with a number. To resolve the issue of handling missing numerical data, four tests ran on the superbio.ai website for ease of use and repeatability. The tests ran with the same neural networks. In one test, the dataset was exactly as downloaded, aside from the one hot encoding. Another test included only the patients with complete data. One test replaced the numerical NaNs with zeros, and the last test replaced the NaNs with the median value of that variable. The MedNan dataset was used for the rest of the experiment. The majority of experiments were conducted using the first 325 patients as a training set and the remaining 87 patients as a validation set. Only when using data selection did the training sets change to either selected data from Data Shapley, randomly selected data, or data with no filled in values that previously were NaNs in numerical features. The validation set was never altered from the definition above.

Exploring Models and Hyperparameter Tuning

Various potential models were explored using the processed data. The first attempt was linear regression. On superbio.ai, a neural network regression and neural network classifier were compared. As the stage variable is a number, a neural network regressor could theoretically be used to predict the stage. Alternatively, a classifier could see the stages as four separate categories. Per the results, using a neural network classifier proved to make more accurate models. In superbio.ai, more tests compared three activation functions between layers – Relu, Tanh, and Sigmoid.

The next experiments ran in a Google Colaboratory (Python 3 environment) notebook to have more flexibility in building models and implementing the Data Shapley algorithm to the learning model. The datasets were loaded into the research notebooks through pandas [9] and rearranged into custom arrays using numpy [10] (both of which are python 3 libraries). The TensorFlow Keras library enabled creating neural network classifiers, which facilitated experimenting with the hyperparameter space to optimize accuracy. To do this, nested 'for' loops tested combinations of three different learning rates and three different regularization rates. These hyperparameters were tested on 18 different architectures ranging from one to five layers and with varying numbers of cells per layer. All the activation functions were Relu, except the last activation function, which was necessarily Softmax per the classification task. Learning rates ranged from .03 to .0003, and regularization rates from .1 to .01. The model that showed the most promise for cirrhosis detection—assumed to have a capacity for learning—could then guide data selection. This model would quantify how much each data point in the data set (patient) contributes to overall learning via a truncated Monte Carlo Data Shapley algorithm.

Why data selection?

Training a model requires a data set, model, and an evaluation metric. Often, a portion of your data set to train your model to accurately predict a pre-defined, separate validation set. When your model is trained to its highest possible accuracy, you can then use it to predict future outputs of previously unseen data. Researchers might want to try randomly selecting training data points or testing on all available data, but data selection seeks to find the most valuable data points for training. The method herein used is called Data Shapley—a data selection algorithm designed by Assistant Professor James Zou at Stanford University.

What is Data Shapley?

Data Shapley is commonly used for supervised machine learning and commonly applied as a Truncated Monte Carlo simulation (TMC). In this research, the algorithm randomly selects one patient from the 418 rows and records the validation loss after training on that data point[4]. Next, another patient is randomly selected, without replacement, and the algorithm will set its Shapley value [11] as the negative of the difference between the resulting loss and the previous loss obtained without that patient in the training set. This process repeats until the loss of our randomly selected training set plateaus. This is one training session, which often includes over 100 random patients in a random ordering, and provides an approximate Shapley value for each patient. After many sessions, there are multiple Shapley values available for each patient, and a 'run' of the Data Shapley algorithm includes a variable number of training sessions as determined by the researcher. This paper takes the mean of all values for a patient, having over 50 approximations per patient. In a case with less than two dozen or so samples of the Shapley value for each patient, the median would be preferable, as outliers may skew the mean. With a value for each data point, Data Shapley allows researchers to select the *n* best data points as the optimal training data.

In the data selection stage of research, each group member independently ran the Data Shapley selection algorithm to generate more approximations for each data point's Shapley value by leveraging the power of multiple computers working in parallel. We combined the individual results using a custom algorithm, which can be found in our GitHub repository. Essentially, the outputs to the individual Data Shapley selection algorithm were a set of patient identification numbers, data_shapley values, and sample counts; we can respectively denote them as PIN, DSV, and SC. PIN is the index number in our original data set downloaded from Kaggle that was assigned to a patient. DSV is the Data Shapley value assigned to that patient—the mean of all approximate shapley values for that patient. SC represents the number of Shapley values obtained for a data point during the duration of a Data Shapley algorithm run.

We combined all Data Shapley record sheets using the following methods (see Code Base in Methods and Materials for the code): The combiner program calculated the weighted mean of the data_shapley values as $Weighted\ mean\ = \frac{\sum_{i=1}^{n} sc_n \cdot dsv_n}{\sum_{i=1}^{n} sc_n}$, where there are *n* separate runs of Data Shapley being combined. The new sample count is set to the sum of all sample counts

[4] In this work, a 'data point' refers to a 'single patient', often referring to the pair of data formed by the input and output values of that patient. The two terms will be used interchangeably.

of that PIN from all Data Shapley output sheets ($\sum_{i=1}^{n} sc_n$). The PIN remains the same and after all calculations the combined output sheet is set to reorder itself numerically based on PINs. The final experiment ran the previously successful architectures with various Data Shapley-selected training datasets, randomly selected datasets, and a dataset of patients that initially had no NaNs in numerical features.

Acknowledgements

This work was made possible with the support and structure of the SHTEM summer internship for high school students. In particular, our collaboration wouldn't have been possible without the work of Sylvia Chin and Professor Tsachy Weissman.

References

1. Karthik Kumar, MBBS. "4 Stages of Cirrhosis of the Liver: 18 Symptoms, Causes & Treatment." *MedicineNet*, MedicineNet, 7 Apr. 2022
2. Superbio.ai. (July 2022). *Machine Learning Workstation*. Custom ML architectures on imported data. https://app.superbio.ai/
3. Nori VS, Hane CA, Sun Y, Crown WH, Bleicher PA (2020) Deep neural network models for identifying incident dementia using claims and EHR datasets. PLoS ONE 15(9): e0236400.
4. Amirata Ghorbani, James Zou. (April 2019). Data Shapley: Equitable Valuation of Data for Machine Learning.
5. TensorFlow Developers. (2022). TensorFlow (v2.8.2). Zenodo.
6. Fleming, T.R. and Harrington, D.P. (1991) Counting Processes and Survival Analysis. Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics, John Wiley and Sons Inc., New York.
7. fedesoriano. (August 2021). Cirrhosis Prediction Dataset.
8. "Cirrhosis." *Mayo Clinic*, Mayo Foundation for Medical Education and Research, 6 Feb. 2021.
9. Jeff Reback, et al. Pandas-dev/pandas: Pandas 1.4.3. v1.4.3, Zenodo, 23 June 2022, p., doi:10.5281/zenodo.6702671 (pandas library)
10. Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau … . (September 2020). Array programming with Numpy.
11. Benedek Rozemberczki, Lauren Watson, Péter Bayer, Hao-Tsung Yang, Olivér Kiss, Sebastian Nilsson, Rik Sarkar. (February 2022). The Shapley Value in Machine Learning.

Figures

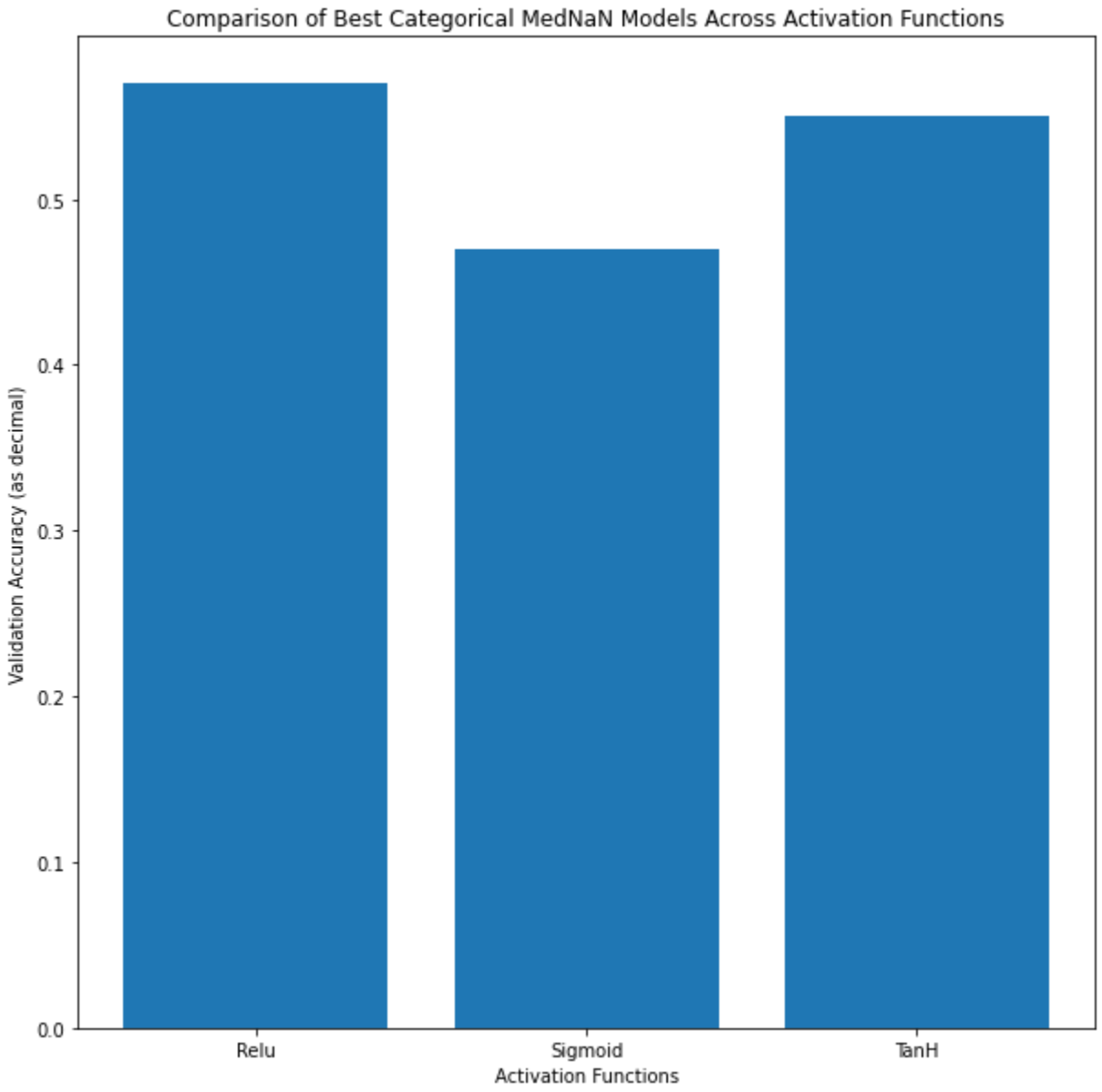

Figure 1: Comparison of Best Categorical MedNaN Models Across Activation Functions. The figure shows the validation accuracy of different activation functions. These experiments ran on superbio.ai with a neural network classifier with varying hyperparameters. The best performance with each activation function is shown (Relu: 4 64 cell layers Learning Rate = .04, TanH: 4 64 cell layers Learning Rate = .0001, Sigmoid 4 64 cell layers Learning Rate = .003).

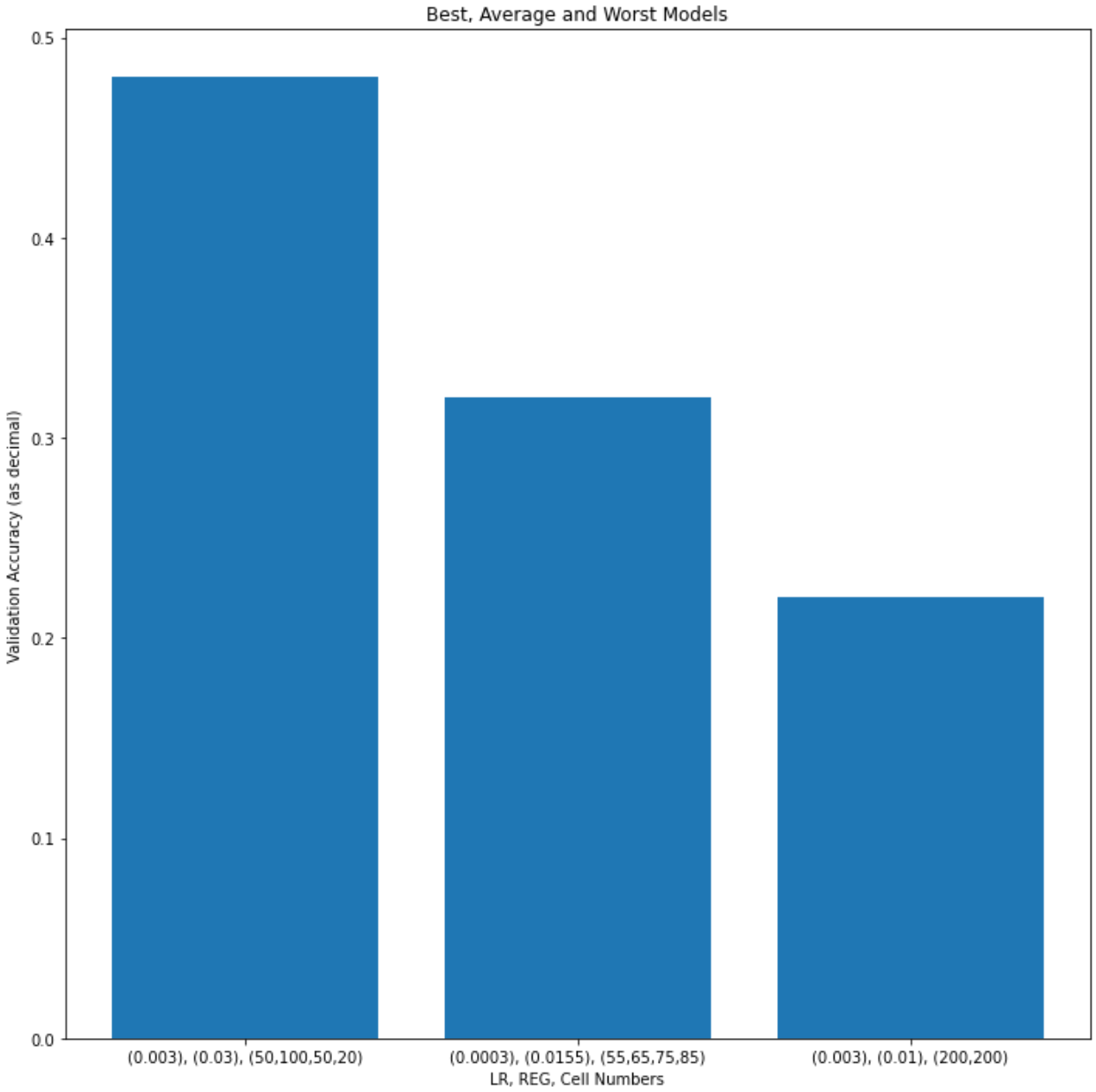

Figure 2: Comparison of best, average, and worst architectures. The figure shows the validation accuracy of the best, average, and worst model when exploring the hyperparameter space. The experiments were performed in a Google Colaboratory environment with the TensorFlow library to explore different architectures, learning rates, and regularization rates.

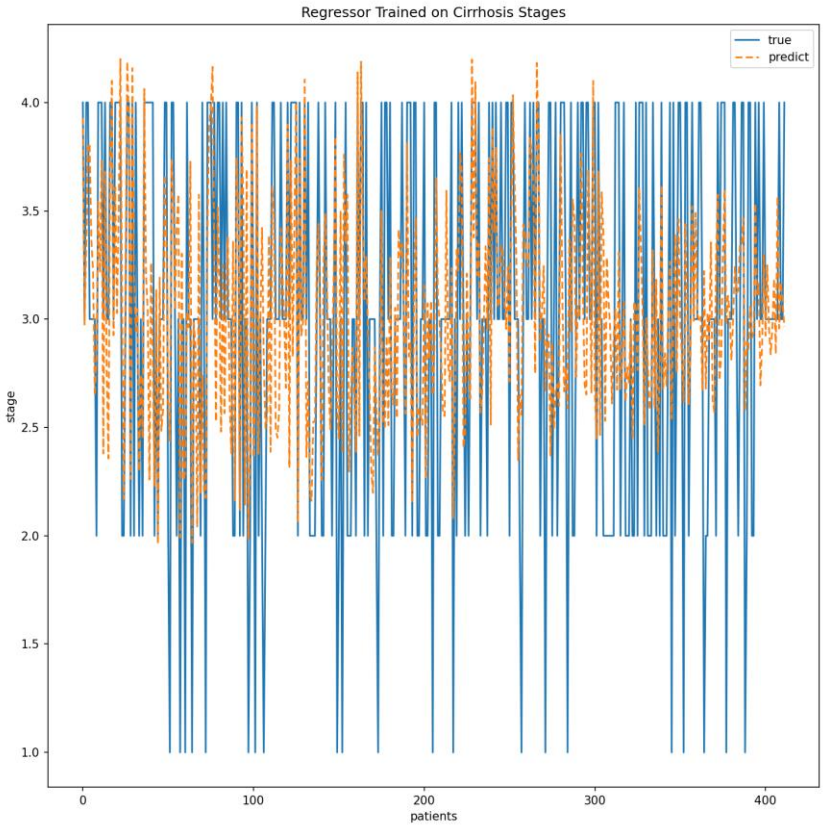

Figure 3: Regressor trained on cirrhosis stages. The plot is of the linear regressions prediction when trained on the data compared to the actual values. This was done in Google Colaboratory with the Keras Tensorflow library.

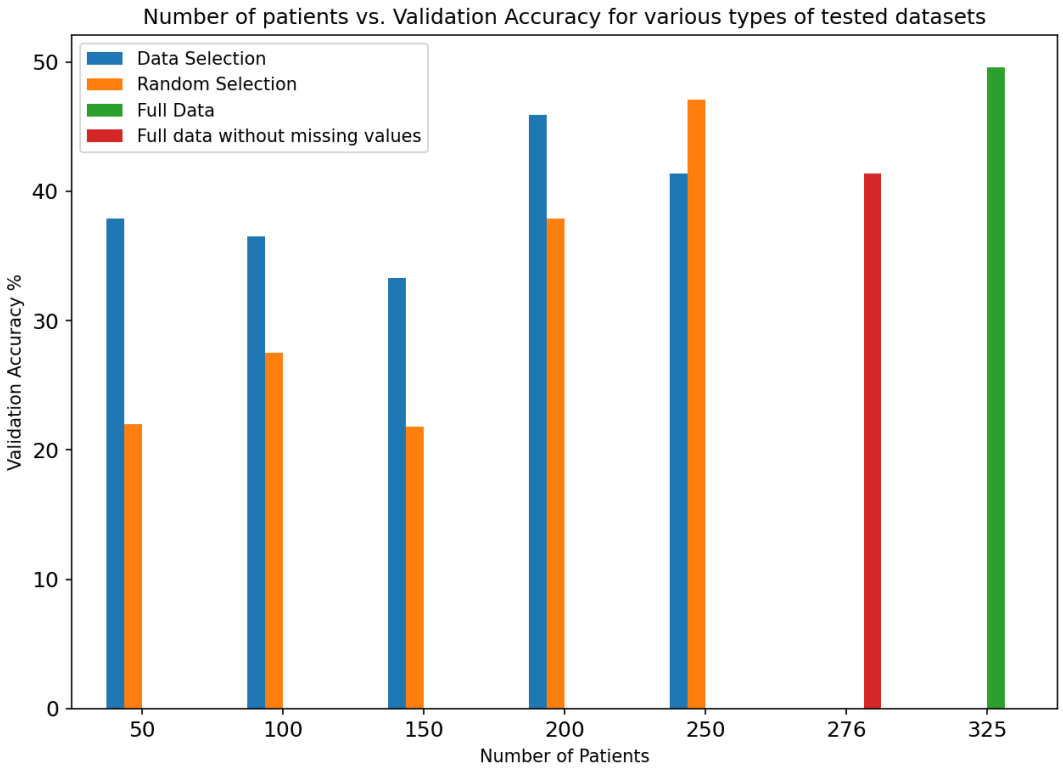

Figure 4: Number of Patients vs. Validation Accuracy for various types of tested datasets. The graph shows the validation accuracy of different subsets of the whole training set, ordered by number of patients. Some are randomly selected, some are selected by the data selection algorithm, one is all the patients with complete data (no missing values), and one is the entire training dataset. All tests used the same validation set. The data selection algorithm was Data Shapley.