Harvard University, CS 109A/209A

Mortality Prediction Using National Health and Nutrition Examination Survey Data

Instructors: Pavlos Protopapas and Natesh Pillai

Rudra Barua, Rosan Bishwakarma, Albert Zhang, Jeremy Zhang

2021 Fall

Contents

1 Mortality Prediction Using National Health and Nutrition Examination Survey Data **3**
1.1 Introduction and Exploratory Data Analysis (EDA) . . . 3
1.1.1 Project Statement . . . 3
1.1.2 Project Goal . . . 3
1.1.3 Description of the Data . . . 3
1.1.4 Visualizations and Noteworthy Preliminary Findings . . . 6
1.2 Data Manipulation and Pre-processing . . . 8
1.2.1 Missing Data Not Missing Completely at Random . . . 8
1.2.2 Missing Data: Multivariate Imputation via IterativeImputer . . . 8
1.2.3 Missing Data: Considerations and Parameters . . . 9
1.2.4 One-Hot Encoding for Categorical Variables . . . 9
1.3 SHAP values . . . 9
1.4 Baseline Model . . . 10
1.5 Decision Tree . . . 10
1.5.1 Single Tree . . . 10
1.5.2 Random Forest . . . 11
1.6 Neural Networks . . . 12
1.6.1 Background . . . 12
1.6.2 Implementation . . . 13
1.6.3 Results . . . 14
1.7 Conclusion . . . 15
1.7.1 Findings . . . 15
1.7.2 Room for Future Work . . . 15

Bibliography **17**

1 Mortality Prediction Using National Health and Nutrition Examination Survey Data

1.1 Introduction and Exploratory Data Analysis (EDA)

1.1.1 Project Statement

What are the factors that increase the risk of someone dying, and how relatively important are each of those factors? While there are many such factors, this project will concentrate on a subset of the factors included in the NHANES I Epidemiologic Follow-up Study. NHEFS is a national longitudinal study that was jointly initiated by the National Center for Health Statistics and the National Institute on Aging in collaboration with other agencies of the Public Health Service. The NHEFS was designed to investigate the relationships between clinical, nutritional, and behavioral factors assessed in the first National Health and Nutrition Examination Survey (NHANES I) and subsequent morbidity, mortality, and hospital utilization, as well as changes in risk factors, functional limitation, and institutionalization.

1.1.2 Project Goal

Initially, we hope to analyze the factors that contribute to mortality in order to build and evaluate a data-driven model for predicting a person's risk of dying based on a set of clinical data and biochemical measurements. We will then try to interpret the model's decisions by looking at how each of the features contributed to the outcome of the model.

The results of our project will be helpful to many parties in society. For example, being able to better predict mortality will help doctors and their patients, healthcare and health insurance companies, scientists, life insurance companies, and more. For the individual, these insight could be helpful in informing lifestyle choices that lead to a healthier life.

1.1.3 Description of the Data

Since the data for NHANES provided on the site is in an old data tapes format, we were provided with a CSV of the data to use (thank you Patrick and Angela for this!). As a result, we have more data from these tapes than was provided in the csv for previous years to students,

but the data is also more raw. Thus, we spent more more time preprocessing and making our own decisions with the data, as you will see below.

What type of data are we dealing with?

The CSV we are using as our sole source of data has 42 columns, the first 41 of which are predictors that we use to predict the target variable y, and 14407 rows of data, one row per subject in the study. All the predictors are from the initial examination that took place from 1971-1975, and, as mentioned in the NHANES documentation, there was no physical examination done during any of the follow ups.

The target variable y for a subject is the time in years that the subject survived since the initial examination till their death. Subjects who were alive at the final 1992 followup or were lost track of before the end of the study have a negative y value in years corresponding to the time between their initial examination and their last follow up (meaning they lived at least this long). A positive y means confirmed death during the study period and that subject died y years after the initial examination, while a negative y essentially means that the subject lived at least as long as abs(y), the absolute value of y, years after the initial examination. Thus, the sign of y just indicates whether we had a confirmed death (positive) or unconfirmed death (negative, in which case we only know the lower bound of how many more years the subject lived).

Here is information on all the 42 columns in our dataset. The first 41 are our predictors.

Column	Data Type	Non-Null Count
sequence_ID	int64	14407
sex_isFemale	bool	14407
race	int64	14407
poverty_index	float64	11348
age	int64	14407
serum_albumin	float64	10801
alkaline_phosphatase	float64	6364
SGOT	float64	6315
BUN	float64	2935
calcium	float64	6256
creatinine	float64	2591
potassium	float64	2992
sodium	float64	3003
total_bilirubin	float64	5854
serum_protein	float64	10800
red_blood_cells	float64	10513
white_blood_cells	float64	12959
hemoglobin	float64	13373
hematocrit	float64	13631

platelets_estimate	object	5850
segmented_neutrophils	float64	5854
lymphocytes	float64	5854
monocytes	float64	5854
eosinophils	float64	5854
basophils	float64	5854
band_neutrophils	float64	5854
cholesterol	float64	13970
serum_iron	float64	9943
serum_magnesium	float64	13814
total_iron_binding_capacity	float64	10149
tranferrin_saturation	float64	9932
urine_albumin	object	13681
urine_glucose	object	13679
urine_pH	float64	13695
urine_hematest	object	13615
sedimentation_rate	float64	10308
uric_acid	float64	6651
systolic_blood_pressure	float64	14339
pulse_pressure	float64	14335
weight	float64	14338
height	float64	14361
y	float64	14407

For the Non-Null Count, the maximum possible value is 14407, since there are 14407 subjects. Notably, the Non-Null Count for y is 14407.

The vast majority of our predictors are numerical: all of the predictors with data type float64 are numerical, plus the age predictor, which is type int64. Notably, our target variable y is a numerical variable. The rest of the predictors, which are the int64 type predictors excluding age and the object type predictors, are categorical.

We used many other methods for exploring the data as well, which we describe in the following subsections.

What are we predicting?

As mentioned above, the response variable can be negative, but a negative value does not indicate that the response variable is actually negative. Rather, the negative sign encodes for the fact that the patient was not confirmed to be deceased during the study period. There are a few options for predictions: predict the raw values of y, predict abs(y), or turn the problem into a classification problem whereby those who survived greater than some number of years was one class and everyone else was another class.

Predicting the absolute value of y initially seemed like a great idea. However, we realized that by doing so, we would be discarding data since the negative/positive encoding would no

longer be present. Turning this into a classification problem also seemed like a great idea. However, we wanted to consider the real world applications of our model. For a doctor or a life insurance company, telling someone that they would live greater than or less than a certain number of years yields very little useful information. For example, for someone in extremely poor health, telling them that they would live less than 20 years is basically stating the obvious.

Given all of these drawbacks, we decided to predict the original value of y.

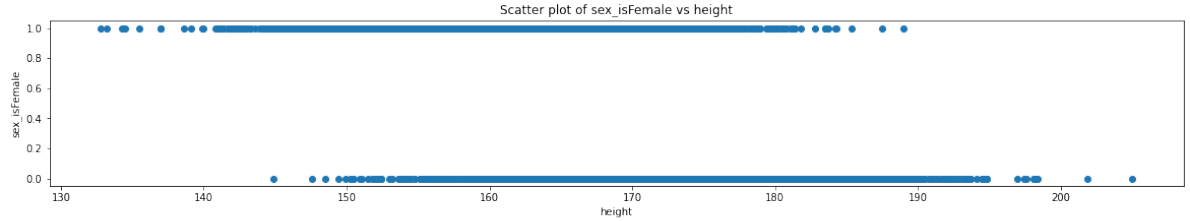
1.1.4 Visualizations and Noteworthy Preliminary Findings

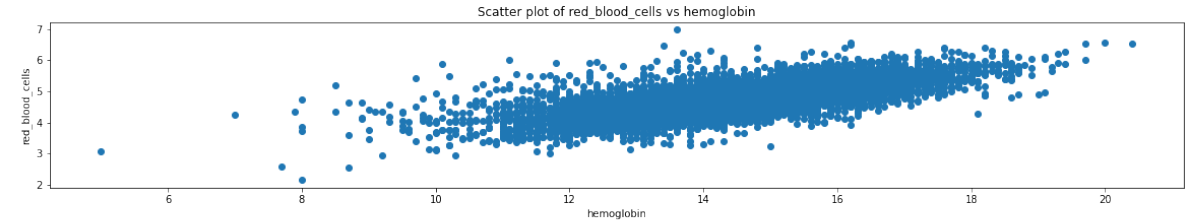
Strong correlation can observed for predictors which one would expect to correlate, either by definition or due to an underlying medical link.

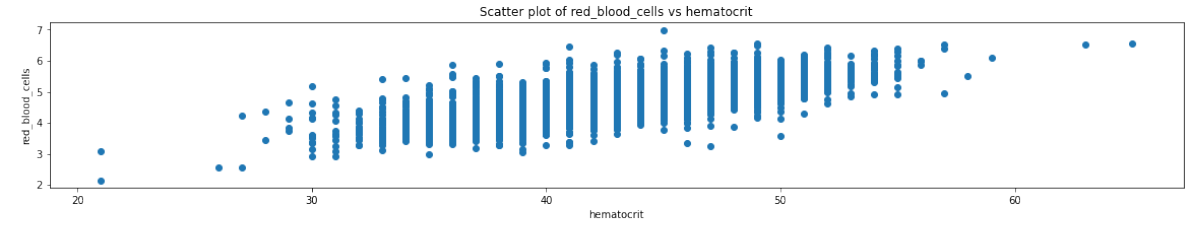
For instance, given that "hemoglobin is a protein your red blood cells"[1], one would expect 'red_blood_cell' to positively correlate with hemoglobin. According to the Red Cross, "Hematocrit level is simply the percentage of red cells in your blood"[2]. It is therefore reasonable to assume that this would correlate with hemoglobin and red blood cell count as well, which the data shows.

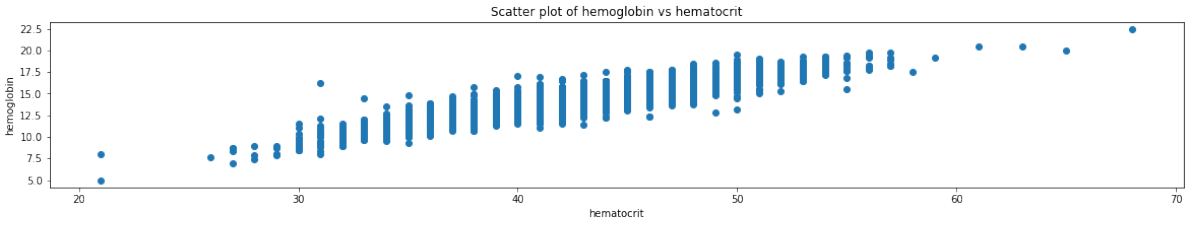
A similar story holds for 'tranferrin_saturation' and 'serum_iron'[3] as well as 'lymphocytes' and 'segmented_neurophilis'[4]. The difference in height by gender was already mentioned above. 'Pulse_pressure' and 'systolic_blood_pressure' are definitionally linked as 'pulse_pressure' is simply the difference between systolic and diastolic pressure[5].

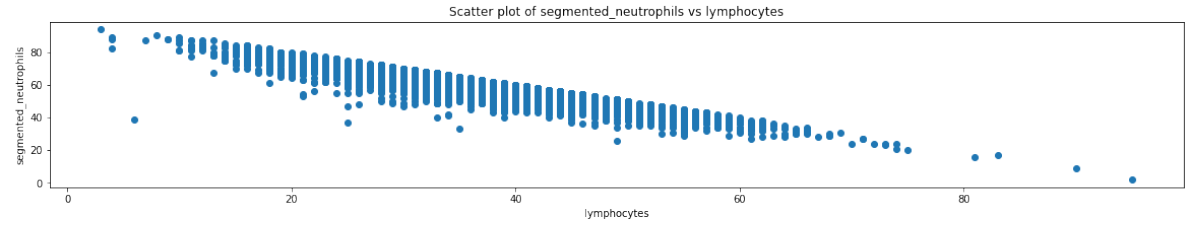
[1] https://www.mayoclinic.org/tests-procedures/hemoglobin-test/about/pac-20385075
[2] https://www.redcrossblood.org/donate-blood/dlp/hematocrit.html
[3] https://emedicine.medscape.com/article/2087960-overview
[4] https://www.urmc.rochester.edu/encyclopedia/content.aspx?ContentID=35ContentTypeID=160
[5] https://www.mayoclinic.org/diseases-conditions/high-blood-pressure/expert-answers/pulse-pressure/faq-20058189

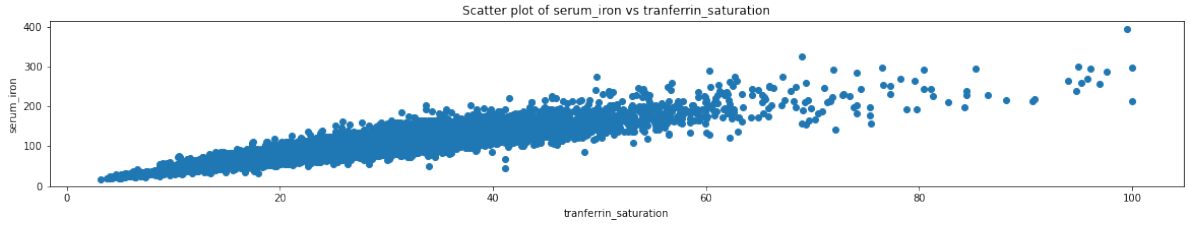

Scatter plot of sex_isFemale vs height
sex_isFemale
height

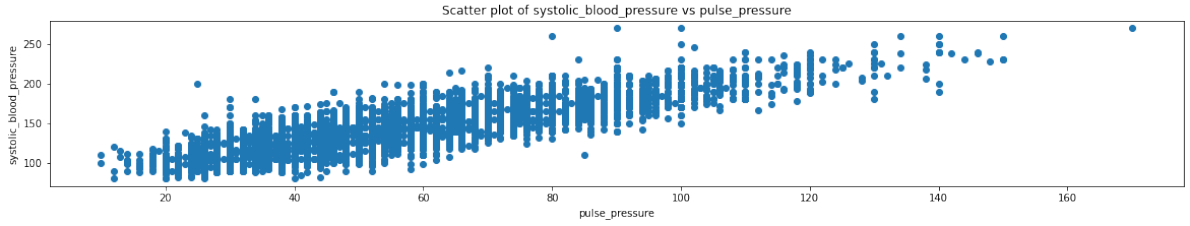

Scatter plot of red_blood_cells vs hemoglobin
red_blood_cells
hemoglobin


Scatter plot of red_blood_cells vs hematocrit
red_blood_cells
hematocrit


Scatter plot of hemoglobin vs hematocrit
hemoglobin
hematocrit

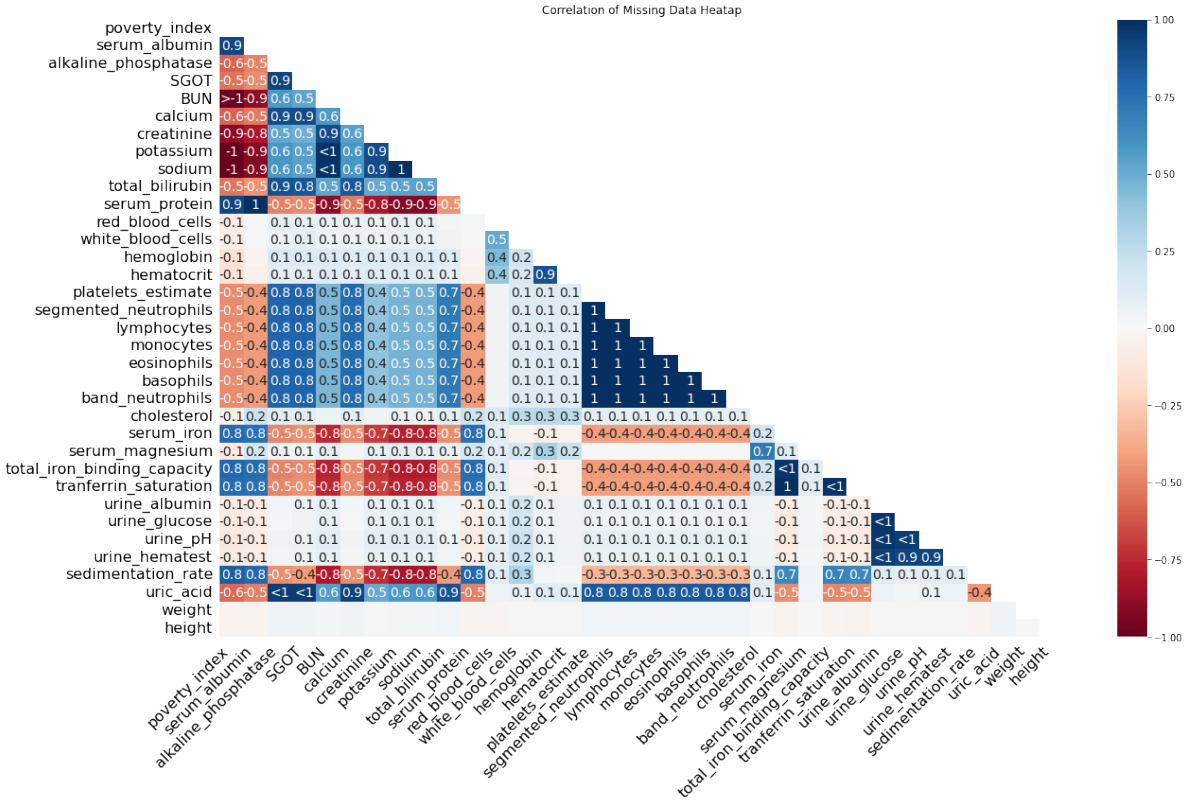

Scatter plot of segmented_neutrophils vs lymphocytes
segmented_neutrophils
lymphocytes


Scatter plot of serum_iron vs tranferrin_saturation
serum_iron
tranferrin_saturation


Scatter plot of systolic_blood_pressure vs pulse_pressure
systolic_blood_pressure
pulse_pressure

1.2 Data Manipulation and Pre-processing

1.2.1 Missing Data Not Missing Completely at Random



Visualizing the correlation between missing values, we can see that there is high correlation between a certain cluster of predictors - 'segmented_neutrophils', 'lymphocytes', 'monocytes', 'eosinophils', 'basophils', 'band_neutorphils' - one hypothesis is that the tests used to obtain these values were run together on the same blood sample.

No matter what the reasons are, we can conclude that the data is not missing completely at random, suggesting a multivariate imputation approach, where each feature is modeled as a function of the other features, e.g. a regression problem where missing values are predicted."[6].

1.2.2 Missing Data: Multivariate Imputation via IterativeImputer

The Multivariate imputation approach allows us make use of the patterns of missing data. This has advantages over a univariate imputation method, which does not make use of those patterns.

Our choice to implement this imputation was scikit-learn's IterativeImputer. The way the imputer works is that each feature with missing values is seen as a function of other features,

[6]https://machinelearningmastery.com/iterative-imputation-for-missing-values-in-machine-learning/

and that estimate is used for imputation. The imputation starts with the features that have the fewest missing values and ascends from there.

It does so in an iterated round-robin fashion: at each step, a feature column is designated as output y and the other feature columns are treated as inputs X. A regressor is fit on (X, y) for known y. Then, the regressor is used to predict the missing values of y. This is done for each feature in an iterative fashion, and then is repeated for max_iter imputation rounds. The results of the final imputation round are returned.

1.2.3 Missing Data: Considerations and Parameters

Since IterativeImputer uses a regression to impute values, we could run the risk of imputing impossible values, such as negative values. After all, one can't be -3 years old. In order to account for this, the min_value parameter was set to zero. This prevents the imputer from imputing negative values.

Moreover, the estimator used was the BayesianRidge as it is computationally more efficient while still performing well according to scikit.learn [7].

1.2.4 One-Hot Encoding for Categorical Variables

In order to capture potential differences between categorical variables, we used one-hot encoding on the following predictors: 'sex_isFemale', 'race', 'platelets_estimate', 'urine_albumin', 'urine_glucose', 'urine_hematest'.

We would expect this to improve predictions of our model as it would be able to learn from these categories as well.

1.3 SHAP values

According to 'Towards Data Science', SHAP values are a derivative of a game theoretic concept that is now used to improve interpretability of models.[8]

Given the difficulty of interpreting models, such as the Random Forest, and the Neural Net, SHAP values were used.

In essence, SHAP values are used to calculate the contribution of a certain feature to the model. In order to achieve that, each possible combination of features is used to train a model, giving 2^F models where F is the number of features. Then, the marginal contributions are weighted and added.

This way, we can ascertain how much a particular feature contributes to the prediction overall.

[7] https://scikit-learn.org/stable/auto$_e$*xamples/impute/plot$_i$terative$_i$mputer$_v$ariants$_c$omparison.html*
[8] https://towardsdatascience.com/shap-explained-the-way-i-wish-someone-explained-it-to-me-ab81cc69ef30

1.4 Baseline Model

MultipleLinearRegression was chosen as the baseline model due to its simplicity. It simple and low cost, and provides interpretable results. Moreover, the baseline model gives us a general direction of what to except from succeeding models, making it a suitable benchmark. More sophisticated models should (in theory) be able to outperform it.

Response Variable	y
Train MSE	136.6647
Test MSE	135.7133
Train R-squared	0.3429
Test R-squared	-0.3367

Table 1.2: scores obtained from Baseline

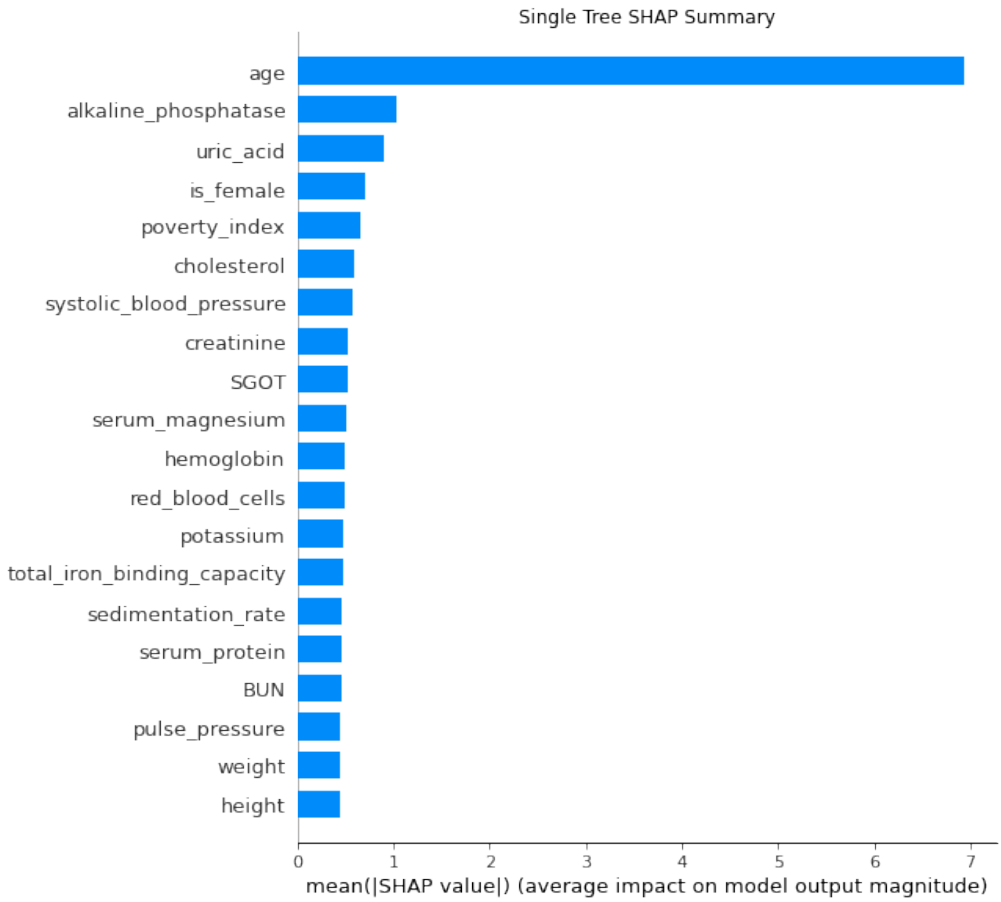
1.5 Decision Tree

1.5.1 Single Tree

Next, a decision tree model with depth = 10 was used to improve on the prediction compared to the baseline model. After scaling our regressors, we obtain the following train and test MSE scores as well as the respective R-squared values:

Response Variable	y
Train MSE	90.7401
Test MSE	172.1048
Train R-squared	0.5637
Test R-squared	0.1589

Table 1.3: scores obtained from Decision Tree



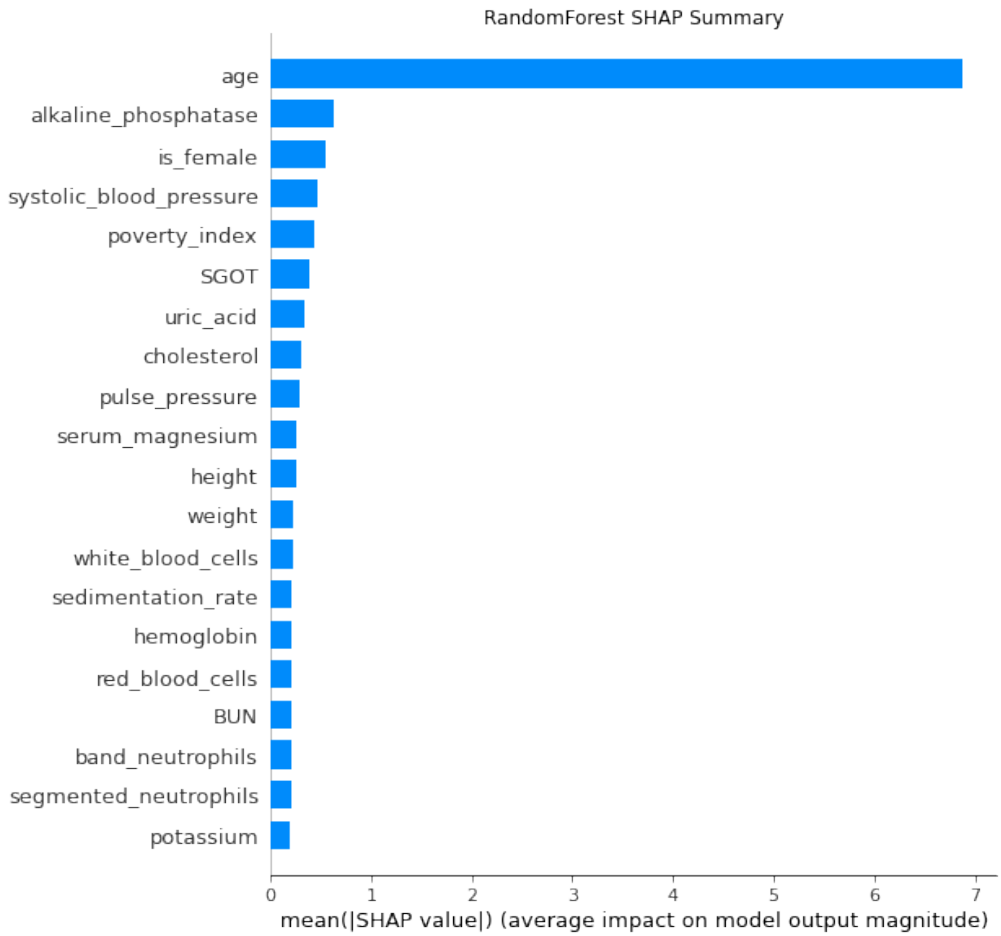
This SHAP graph indicates that ‘age‘ is the most important predictor by far in terms of average impact on model output magnitude, followed by ‘alkaline_phosphatase‘ and ‘uric_acid‘.

1.5.2 Random Forest

To further improve upon the model, we implemented a Random Forest.

Response Variable	y
Train MSE	21.4560
Test MSE	136.2687
Train R-squared	0.8968
Test R-squared	0.3340

Table 1.4: scores obtained from Random Forest



This SHAP graph indicates that ‘age‘ is the most important predictor by far in terms of average impact on model output magnitude, followed by ‘alkaline_phosphatase‘ and ‘is_female‘.

1.6 Neural Networks

1.6.1 Background

As a note, the authors of this paper learned neural networks using Kaelbling (2020), Deuschle (2021), Liang and Sadigh (2020), and Ng (2020), including learning the theory behind them, their advantages and disadvantages, how to implement them in code, etc., so much of the concepts and information throughout this chapter comes indirectly from these sources. Thus, to avoid clutter, rather than repeatedly cite these sources every sentence, we are just saying here that we use these sources throughout this chapter. We also recommend interested readers check out the above sources, as they really are quite interesting and helpful for learning more about neural networks - they explain the concepts of layers, activation functions, deepness, etc. for neural networks.

We decided to implement neural networks as our method that was not discussed in class. Our rationale for choosing to try neural networks as a model is that neural networks, compared

to linear or polynomial regression, are able to be more expressive since they can better model non-linear and more complex relationships through using multiple layers and activation functions. Through our EDA and our previous results, we've seen how the relationship between our predictors and our target variable is indeed quite complex, so using neural networks to try and better capture that complexity seemed like a good direction to explore.

One caveat is, neural networks are not that interpretable, especially when they are deep, since they have so many parameters, and parameters in each layer do not have a convenient interpretation because of the way that parameters affect subsequent layers in increasing complexity. Neural networks would likely be the least interpretable model we use, except for perhaps Random Forests or other ensemble methods. To combat this issue of worse interpretability, we used SHAP (Shapley Additive exPlanations) to improve our interpretability, which we discuss elsewhere in our report. Another caveat is that our previous results suggest that age is the most dominant predictor, so this may restrict our neural networks performance since, if age is the most helpful predictor by far and the other predictors are not that helpful, then the full potential expressiveness of our neural network won't be utilized since age would just dominate the model.

1.6.2 Implementation

To code our neural networks, we used the TensorFlow library so we could focus on the high level aspects of the neural network implementation and not worry about the nitty-gritty details. We create the function `deepNNet(X_train, X_test, y_train, y_test, epochs)` that first normalizes the given training and test data, then creates a feed-forward/sequential neural network model with three layers using the ReLU (Rectified Linear Units) activation function, which provides non-linearity in our neural network since ReLU(x) = $\max(0, x)$. We have 53 neurons in each of the first two layers, which strikes a good balance of our neural network being expressive while not too complex, since complexity would lead to overfitting and longer training times. Our last layer just has 1 neuron since our output (target predictor) is one-dimensional. We decided to do this all in a function `deepNNet` so that we can run this function on the different sets of training and test data that we have, like using y vs abs(y) as the target variable, or splitting the datasets into positive y and negative y cases.

We use Adam as our optimizer, which is an adaptive learning rate optimization algorithm for stochastic gradient descent for training deep learning models designed specifically for training deep neural networks from Kingma and Ba (2017). Using Adam helps our neural network to be trained faster, especially since Adam is particularly well suited for large amounts of data and parameters, which we have in our case. For the loss function, we use MSE since we are still dealing with a regression problem.

Then, we train our neural network using the training data and validation data for 100 epochs, where in one epoch the neural network is trained on all of the training data exactly once. We chose 100 based on some testing, since performance of our neural networks did not seem to improve much for more epochs, and more epochs requires more training time, so 100 was a good balance between performance and training speed. Each epoch consists of multiple

batches, and we set the batch size to be 32, where a batch consists of a set of training samples that our neural network processes before updating its parameters, so our neural network processes 32 training samples before updating parameters according to our Adam optimizer. We similarly chose 32 because it struck a good balance between performance and training speed.

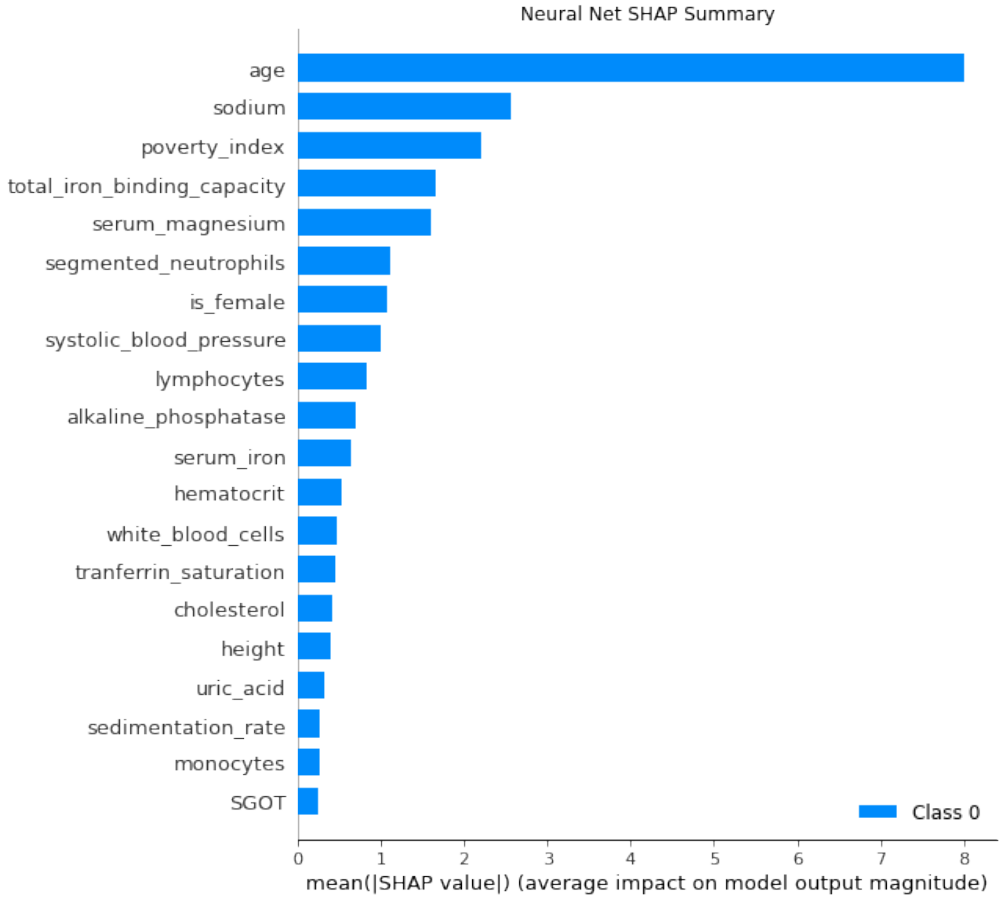
1.6.3 Results

We ran our `deepNNet` function for the different sets of training and test data that we have.

For using y as our target variable, our neural network achieved

Train MSE	130.2074
Test MSE	127.7878

For our SHAP values, we obtained



which indicates that ‘age‘ is the most important predictor by far in terms of average impact on model output magnitude, followed by ‘sodium‘ and ‘poverty_index‘.

Overall, our neural networks achieved good performance relative to our models but was not overwhelmingly better. So, it is not clear whether using neural networks is the best choice because there is the trade-off of slightly better performance at the cost of interpretability. Depending on the priorities, it could be better to use neural networks, or it could be better to use one of our other models.

1.7 Conclusion

1.7.1 Findings

Looking at the test MSE for all of the different models, we see that the baseline model had a test MSE of 135.7133, the single decision tree had a test MSE of 172.1048, the random forest had a test MSE of 136.2687, and the neural network had a test MSE of 127.7878.

In terms of using the MSE to evaluate model performance, we see that the neural network performed the best since it has the lowest MSE score. As expected, the random forest had better performance than the single decision tree, but the random forest had very similar performance to our simple baseline model. Overall, while the neural network did perform the best, we see that it does not do significantly better than the other models. Despite the neural network performing the best, we would probably not recommend it for use in real world applications. We envision that those involved in the medical community such as doctors, researchers, and life insurance companies will use this algorithm to predict mortality. Often, these stakeholders will want a degree of interpretability in the models to inform their decisions or their patients, which the baseline model and single decision tree all provide while the neural network and random forest are less interpretable.

Using our SHAP graphs, we see that for all of our models, ‘age‘ was the most important predictor in terms of average impact on model output magnitude. This makes sense since older people are likely to have shorter remaining lifespan, with how old someone is and their remaining lifespan being very inversely correlated. For the models, the SHAP values for the other predictors varied, with the neural network model having the highest SHAP values for predictors other than ‘age‘. This makes sense since the complex and expressive nature of the neural network enabled it to more effectively utilize the other features, giving features other than ‘age‘ higher SHAP values.

1.7.2 Room for Future Work

In the future, one could expand upon our work with neural networks by experimenting with more complex types of neural networks, such as Recurrent Neural Networks (RNNs) or Convolutional Neural Networks (CNNs), rather than the feed-forward neural networks we used, though from our understand RNNs and CNNs don’t seem to match the structure of our data well, which is why we did not use them (since RNNs usually are better for sequential data and CNNs are better for image or video data). One could also expand upon our work by trying different optimizers other than Adam. Furthermore, in the future, we would like to

explore the Cox's proportional hazards model. Although less complex than a neural network, it is considered a popular survival model because of its interpretability. There are a ton of interesting models to experiment with in the realm of neural networks, and new research is being done in this area all the time, so there are many avenues for expanding upon our work.

Bibliography

Deuschle, W. J. (2021, April). Undergraduate fundamentals of machine learning. https://github.com/harvard-ml-courses/cs181-textbook/blob/master/Textbook.pdf. Harvard University, CS 181 Textbook.

Kaelbling, L. (2020, Fall). Course textbook for MIT 6.036. https://introml.odl.mit.edu/cat-soop/_static/6.036/LectureNotes/6_036_lecture_notes_spring2021.pdf. MIT, 6.036 Introduction to Machine Learning.

Kingma, D. P. and J. Ba (2017). Adam: A method for stochastic optimization.

Liang, P. and D. Sadigh (2020, January). Stanford CS 221: AI (Autumn 2019). https://www.youtube.com/playlist?list=PLoROMvodv4rO1NB9TD4iUZ3qghGEGtqNX. Lectures from Stanford Online on YouTube.

Ng, A. (2020, April). Stanford CS 229: Machine Learning (Autumn 2018). https://www.youtube.com/watch?v=jGwO_UgTS7I&list=PLoROMvodv4rMiGQp3WXShtMGgzqpfVfbU. Lectures from Stanford Online on YouTube.