Names and NetIDs:
Alan Xia (ax2054)
Matt Grzyb (mdg9707)
Ben Davis (bfd233)
Ajibike Lawal (asl613)

Data Page:
https://www.kaggle.com/andrewmvd/heart-failure-clinical-data

1. INITIAL REACTIONS AND DATA ORGANIZATION

1.1 INTRODUCTION

Cardiovascular diseases (CVDs) such as coronary heart disease, cerebrovascular disease, and peripheral arterial disease are the leading cause of death globally. CVDs take an estimated 17.9 million lives per year and account for 31% of deaths worldwide. Most CVDs are preventable by addressing certain actions such as smoking, unhealthy diets, physical inactivity, and excessive consumption of alcohol. The dataset provided gives us a glimpse of what we could use to predict the likelihood of death.

In our analysis, we investigated three main questions:
1. What is the best model to serve as a predictor for death for those with heart disease?
2. To what extent do the levels of serum sodium in a high-risk individual correlate with death?
3. Does serum creatinine or ejection fraction correlate better with death?

Using the data, we were interested to see the effects of various parameters on the likelihood of death and we knew maintaining accurate results would be important in interpreting the data. Our initial reaction of the dataset was that there would be obvious parameters that lead to death such as age, sex, and time since the last visit. However, we were more interested in seeing if the alarming results of specific tests would correlate with the death of the patient.

1.2 DATA ORGANIZATION AND FILTRATION

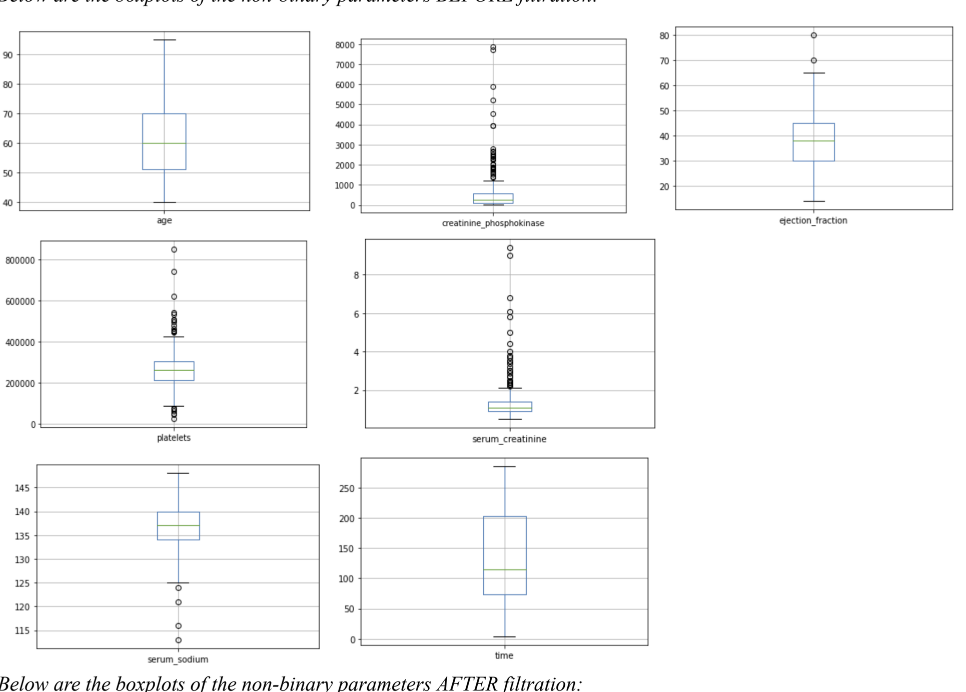
To ensure that the proceeding results are as accurate as possible, we had to first filter the data for outliers and null values, check for imbalances, and determine if the changes made to the data are reliable and significant.

```
age                         0
anaemia                     0
creatinine_phosphokinase    0
diabetes                    0
ejection_fraction           0
high_blood_pressure         0
platelets                   0
serum_creatinine            0
serum_sodium                0
sex                         0
smoking                     0
time                        0
DEATH_EVENT                 0
dtype: int64
```

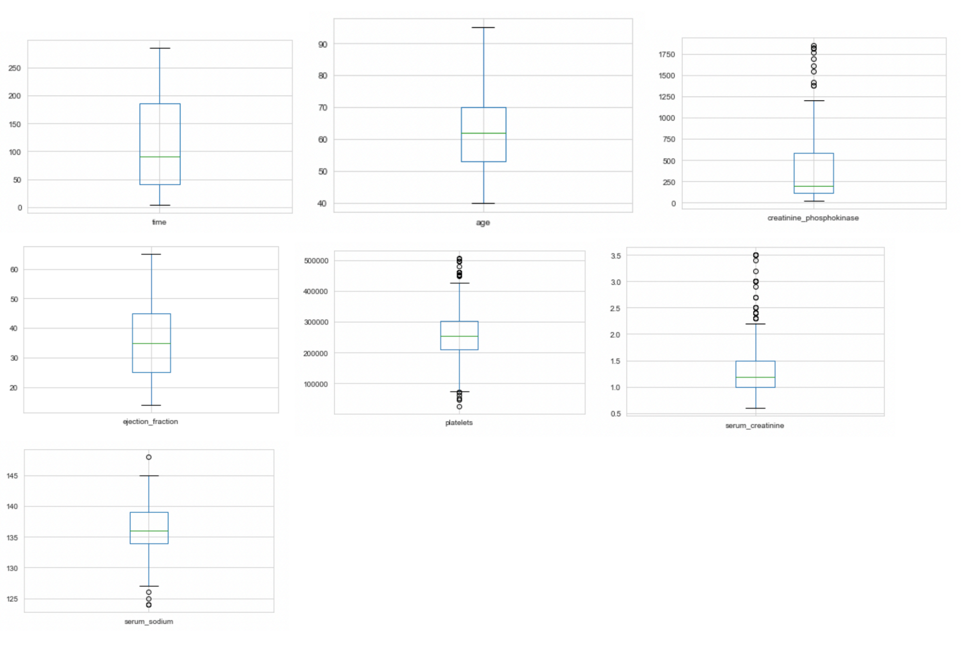
When we checked the data for null values we found that there were no nulls.

Before we checked if there were any imbalances, we first had to ensure that there were no outliers negatively impacting our results. We decided that the best method in reducing outliers for most of the parameters was to use the normal distribution and z-scores. Assuming that the criteria of the Central Limit Theorem is met, we can say that any value with a z-score of three or greater would be considered an outlier. However, we had to be careful in marking specific values as outliers since certain parameters had ranges of values that were acceptable and data were being incorrectly marked as outliers. For certain parameters, we corrected any mistakes in the marking of outliers by hand to ensure the validity of the data filtration.

Below are the boxplots of the non-binary parameters BEFORE filtration:



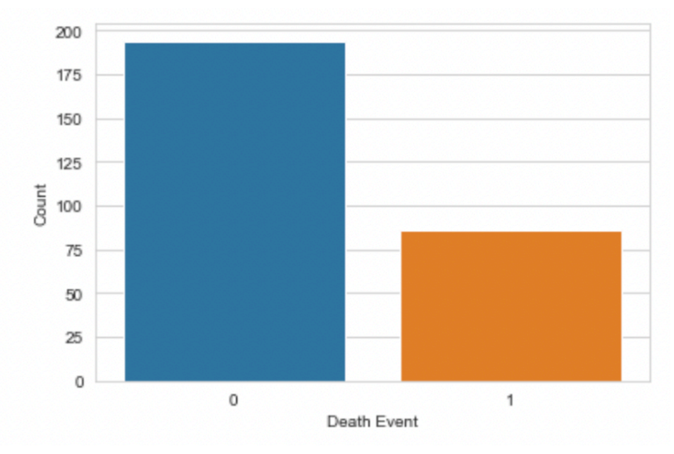
Below are the boxplots of the non-binary parameters AFTER filtration:



As seen, even after the filtration of outliers of the non-binary parameters, there are still data points that are displayed as outliers by the software. However, we deemed the remaining data points as valid data since they are within the reasonable range for their parameter.

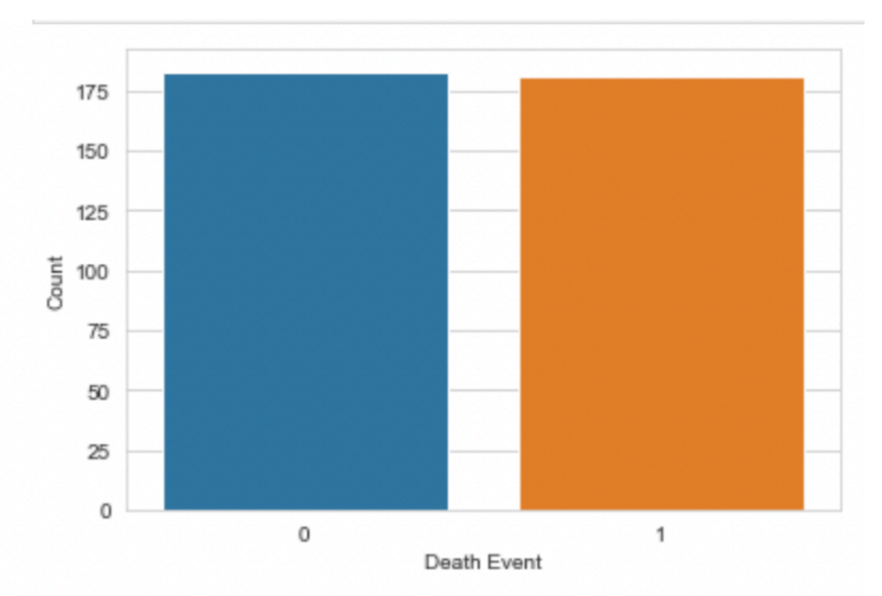
We noticed that there was a severe class imbalance in the death event parameter. In order to prevent high accuracy scores in the machine learning models, we decided to upsample the minority class to rid the data of any imbalance.

Below is a graph of the imbalance of the death event parameter BEFORE upsampling:



As seen, there is a severe imbalance in the minority class (death). To reduce class imbalances in the dataset, we decided to utilize upsampling for the minority class. Before we upsample the data, we created a logistic regression model so we could compare to it after we upsampled the data to see if the results were significant. After running the logistic regression model, we found the accuracy of the model to be 75.2%.

Below is a graph of the imbalance of the death event parameter AFTER upsampling:

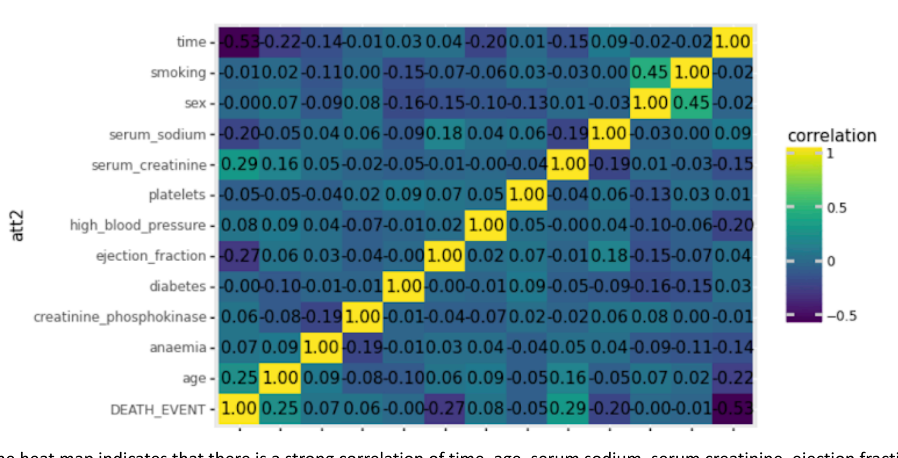


After upsampling the minority class, reducing outliers and checking for null values, the significance of the data filtration and the accuracy of the changes has to be determined. By comparing the accuracy of the same machine learning model before and after the changes, we are able to determine the extent that the changes improved the accuracy of the predictions. After we upsampled the data, the accuracy of the logistic regression model increased to 85.3%. This is a 10.1% increase in accuracy scores of the output for the logistic regression model. Therefore, we can be confident that even though there may be a bias due to upsampling, that the overall accuracy and reliability of the predictions increased.

1.3 VISUALIZING THE CORRELATIONS

Visualizing the correlations of the different parameters to death was important in creating the perfect models with the highest accuracy. Knowing the correlations, we are able to gauge a sense of what data we should expect to see and what parameters could possibly be eliminated due to a lack of influence.

Below is a heat map of the correlations between all parameters:



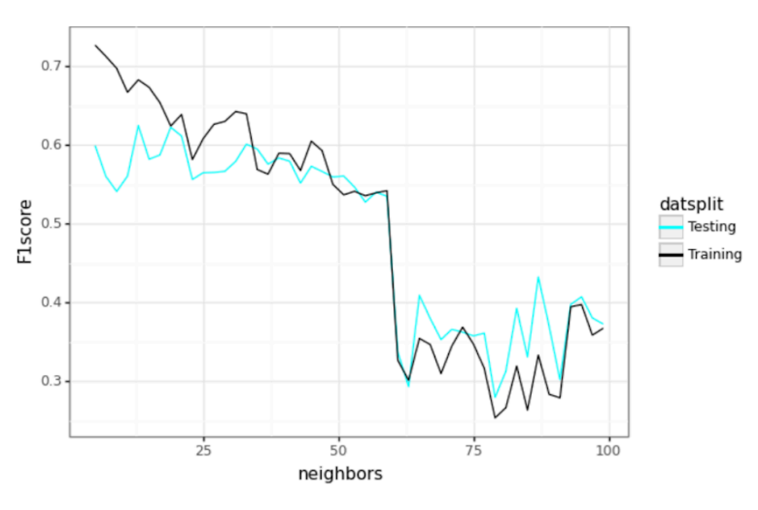
The heat map indicates that there is a strong correlation of time, age, serum sodium, serum creatinine, ejection fraction, and age to death. Certain parameters such as sex and age appear to have a relatively little correlation to death. Having a general idea of what parameters are highly correlated with death is crucial in moving forwards and creating a model with the highest accuracy possible.

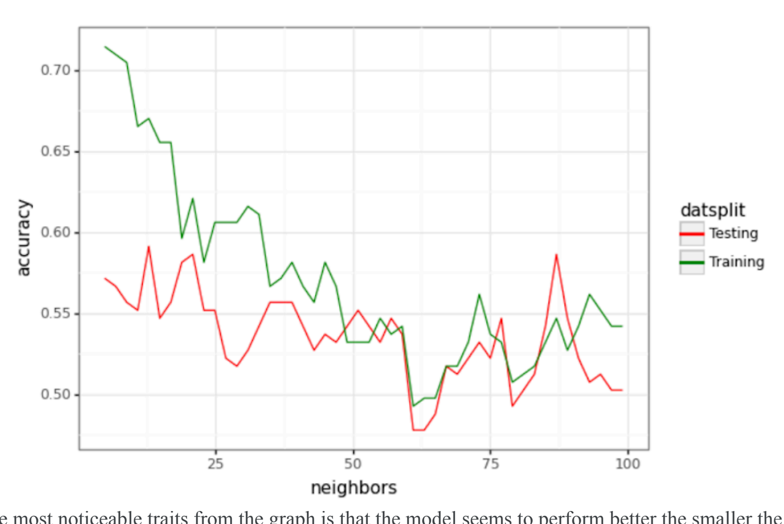
2. SELECTION OF MACHINE LEARNING MODELS

One of our goals in analyzing the dataset was to create effective machine learning models that predicted death based on the variables presented in the data set. One of our first steps in creating the model was deciding on whether to use regression or classification for our machine learning techniques. Generally, regression deals with predicting continuous quantities, that is a function (f) with inputs x maps out a y. Classification on the other hand deals with predicting the probability of certain classes, and in our project these classes would be one death-class and one non-death class.
To start off in our analysis we ran a k-nearest-neighbor classification model.

```
              precision    recall  f1-score   support

           0       0.70      0.76      0.73       102
           1       0.74      0.66      0.70       101

    accuracy                           0.71       203
   macro avg       0.72      0.71      0.71       203
weighted avg       0.72      0.71      0.71       203
```

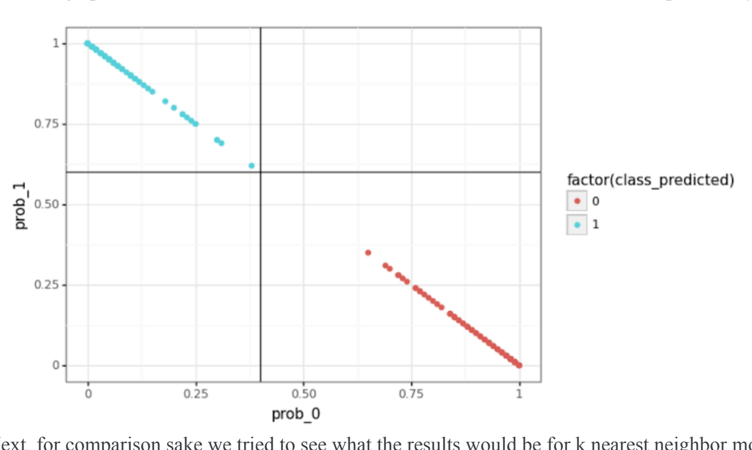
At 71% our accuracy was okay but there definitely is room for improvement.
We got 70% for the true positive rate but unfortunately, our recall with class 1 was only 66%. This model was run using 3 nearest neighbors and the rest of the parameters were set to default. .
In order to better understand the performance of the model, we plotted both our training and testing scores with neighbors on the x axis while plotting the f1 score and accuracy on the y axis respectively.





The most noticeable traits from the graph is that the model seems to perform better the smaller the amount of neighbors.

Here is a graph that shows the threshold the model uses to define classes. Here the probability value is at 0.4.



Next, for comparison sake we tried to see what the results would be for k nearest neighbor model using regression instead. Our results were:
Using this method we were able to get better results and our area under curve score is definitely 'acceptable' since its

```
              precision    recall  f1-score   support

           0       0.81      0.80      0.81        97
           1       0.82      0.83      0.83       106

    accuracy                           0.82       203
   macro avg       0.82      0.82      0.82       203
weighted avg       0.82      0.82      0.82       203
```

0.817 `0.8171561952927446`

Next in order to ensure that the hyperparameters we used are the most effective we implemented grid search strategies that told us what values we should set for our hyperparameters. Generally the parameters for k nearest neighbors involve:
- algorithm
- leaf_size
- metric
- N_jobs
- n_neighbors=5
- p=2
- Weights

We were able to find the print out the best hyperparameters python thought we should use and then we applied it to our model.

```
Best leaf_size: 1
Best p: 1
Best n_neighbors: 1
```

When applied we got :

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
Classification Report :               precision    recall  f1-score   support

           0       0.88      0.81      0.84        36
           1       0.86      0.91      0.88        46

    accuracy                           0.87        82
   macro avg       0.87      0.86      0.86        82
weighted avg       0.87      0.87      0.87        82
```

Our results improved significantly. At 87% accuracy.
We also ran a couple other machine learning techniques by using Random forest and SVM, getting these results respectively.

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
Classification Report :               precision    recall  f1-score   support

           0       0.90      1.00      0.95        36
           1       1.00      0.91      0.95        46

    accuracy                           0.95        82
   macro avg       0.95      0.96      0.95        82
weighted avg       0.96      0.95      0.95        82
```

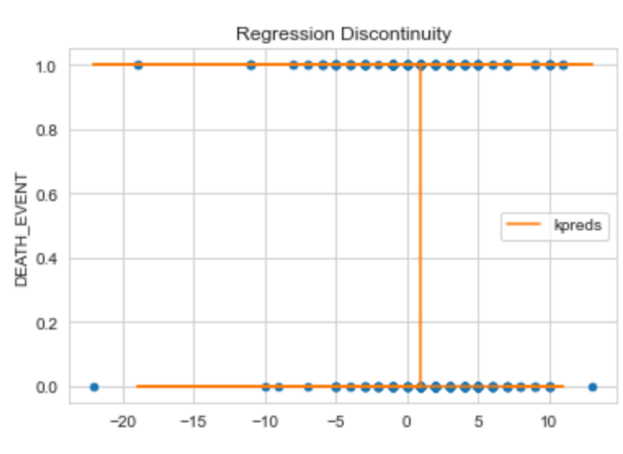
```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
Classification Report :               precision    recall  f1-score   support

           0       0.90      1.00      0.95        36
           1       1.00      0.91      0.95        46

    accuracy                           0.95        82
   macro avg       0.95      0.96      0.95        82
weighted avg       0.96      0.95      0.95        82
```

3. SERUM SODIUM AS A PREDICTOR OF DEATH?

In this project we mainly examined the effect that ejection fraction had on death likeness but we were also interested in finding out the impact that serum sodium has on death. Serum sodium is the measurement used to quantify sodium levels in people's blood. According to the Mayo Clinic the normal/healthy range for serum sodium is 135-147 mmol/L. The mayo clinic interprets any level below 135 as dangerous and a likely indicator that someone suffers from Hyponatremia. In order to explore any causal inferences from having dangerously low sodium levels with death we used regression discontinuity to explore causal relations. To set up the regression we set below 135 as the threshold and using weighted least squares in order to account for the variance among serum sodium in our data. Running the regression we got:

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.2639	0.013	20.536	0.000	0.239	0.289
serum_sodium	-0.0097	0.003	-3.573	0.000	-0.015	-0.004
threshold	0.2639	0.013	20.536	0.000	0.239	0.289
serum_sodium:threshold	-0.0097	0.003	-3.573	0.000	-0.015	-0.004

Here we see that the model tells us that the likelihood of death increases by 0.2369 points with a serum sodium level below 135. Basically, having Hyponatremia or a serum sodium level that is below 135 increases the likelihood of death by 2%. (threshold+intercept)/intercept. The results are significant since the p values are extremely small and practically 0!



Above is the visualization with the yellow line outlining our k nearest neighbor predictions, since we added k-prediction as a column in the data. Unfortunately, we do have to keep in mind that the individuals in the dataset already have cardiovascular diseases so the results are definitely biased.

4 SERUM CREATININE VS EJECTION FRACTION AS PREDICTOR OF DEATH

There are many factors that lead to death for those with heart disease. In this case, we want to focus on serum creatinine and ejection fraction. Our goal is to identify what a normal range looks like for each, and see whether falling into the normal range or not can be used to correlate with death event. We also want to investigate whether serum creatinine or ejection fraction works better, as measured by correlation.

The implication of this work is that it could allow doctors to better predict the outcome of death based on test results from serum creatinine or ejection fraction.

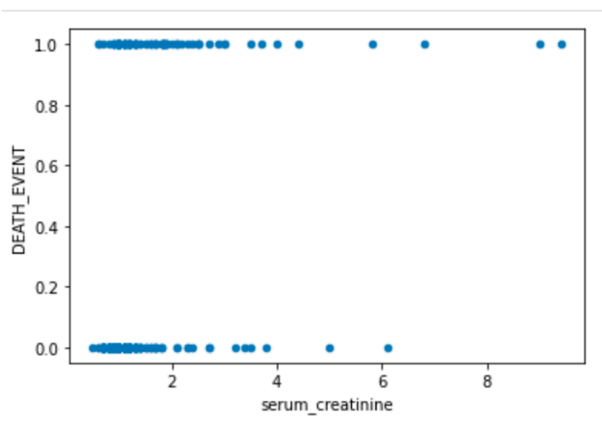
4.1 ESTABLISHING NORMAL RANGES

We created a dataframe using ejection fraction, serum creatinine, sex, and death event, keeping sex as gender is a factor of consideration we use later. According to the Mayo Clinic, normal creatinine levels for are :
- For adult men, 0.74 to 1.35 mg/dL (65.4 to 119.3 micromoles/L)
- For adult women, 0.59 to 1.04 mg/dL (52.2 to 91.9 micromoles/L)
  - Source: https://www.mayoclinic.org/tests-procedures/creatinine-test/about/pac-20384646

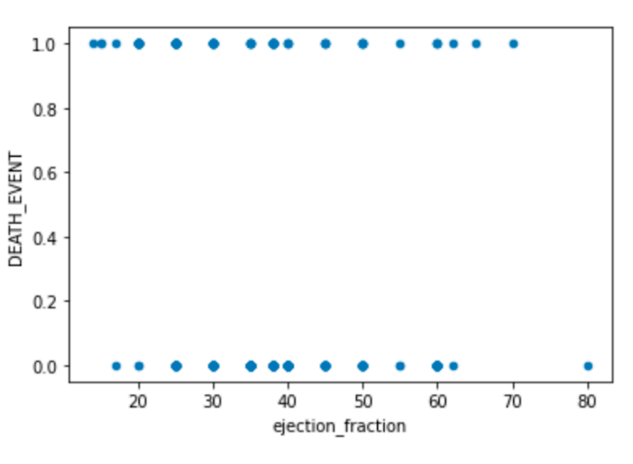
For ejection fractions, the normal range is about 50% to 75%, according to the American Heart Association. A borderline ejection fraction can range between 41% and 50%. Below 40% is considered heart failure.
- Source: https://www.mayoclinic.org/tests-procedures/ekg/expert-answers/ejection-fraction/faq- 20058286

However, for the sake of the test and determining a normal ejection fraction range, we count above 50 as normal.

Then we plot both serum creatinine and ejection fraction against death event to see what we can observe. The death event can be expected to only be 0 or 1 since it is binary data where 1 = death and 0 = no death.



The tight clustering to the left indicates that there is a normal range for serum creatinine. The difference between the top and bottom tell us that there is some correlation between serum creatinine and death as the points plotted at the top are those representing death event and the points at the bottom are those representing no death event.



Again comparing the top and bottom points plotted, it can be observed that the two lines are similar. Indicates that there is a weaker correlation, but we still want to check.

4.2 CORRELATION AND DATA CONVERSION

Now that we have normal and abnormal ranges for both serum creatinine and ejection fraction, we add two new columns to our dataframe (one for normal serum creatinine and another for normal ejection fraction). For both, we use binary values where 1 = normal and 0 = abnormal.

The correlations we receive are represented in statistics by r, or the coefficient of correlation.The correlation between normal serum creatinine and death event is -0.2208. The correlation between normal ejection fraction and death event is -0.1082.

Both have negative correlations which gives us the logical conclusion that abnormal values lead to death events.

However, both correlations are low, which suggests that this would not be an efficient metric. Previous results in part 1 of the final indicate that in tandem with other factors both serum creatinine and ejection fraction are useful, they just cannot be used individually.

As to which one is more effective, the efficacy of both are low but the correlation with normal serum creatinine and death event is better in comparison to that of ejection fraction.