**CSCI 381/780 – Applied Data Science**

**Jacob Weissman Final Report**

# Project Abstract

Myocardial infarction, colloquially referred to as ‘heart attack’, can lead to both short-term and long-term complications, even among patients who receive medical care. Critically, if complications or death are known to be at a high risk of occuring, preventative measures can be taken.

The aim of this project was to build an interpretable model to predict lethal outcomes of patients after three days in the hospital for myocardial infarction. Models were scored based on a weighted F1 formula, where the weights of recall and precision in the F1 formula were left as hyperparameters to medical specialists. This allows each treatment facility to appropriately weigh the availability of resources and the danger of false negatives. During training, F1 scoring was used. During my testing, recall importance was set to 0.5, resulting in F1 also being used for scoring.

I used available features to predict complications. The outcome of this step was a random forest model for each complication. Some complication models, generally for complications that occurred less frequently in the dataset, performed poorly. This could be caused by more samples being needed for robust model performance.

The outputs of the high-performing complication models, along with the original features, were used to predict lethal outcomes. A random forest model was used, which allows medical specialists to view and interpret the feature importance. The model utilizing complications predictions performed similarly on F1 to the model that only utilized features, but the former significantly outperformed the latter on recall.

# Accomplishments

### Major Goals

The major goal was to use machine learning to develop an interpretable model that can predict the risk of death among hospitalized patients who have suffered a myocardial infarction. If the performance was strong and the results were interpretable, the tool could be used by medical specialists to craft care plans. Since my model is being used for prediction on day 3, it will not have access to “complications” information when making predictions. However, I wanted to find a way to leverage the data in “complications” to improve model performance.

### What did you accomplish?

Specific Objectives / Major Activities

**Aim 1: Perform data preprocessing for machine learning**

Aim 1.1: Convert lethal outcome to binary dead / alive

This step involved defining the target variable, ‘LET\_IS’, and using built-in pandas functions to convert all lethal outcome types (1-7) into a single outcome, death (1). I hoped this would make model performance more robust, especially due to the small number of available samples (only 1700 total samples available).

Aim 1.2: Perform stratified split into datasets

Because there were many more ‘alive’ cases than ‘dead’ cases (84.06% alive according to the description attached to the dataset), a stratified split was necessary when dividing the data into datasets. I formed a test dataset with 15% of the data, a validation dataset with 15% of the data, and a training dataset with 50% of the data. I then created two additional datasets, tune and threshold, each with 10% of the data. Tune was created to perform feature selection, and threshold was created to calculate optimal thresholds for the random forest regressor. As a side note, preliminary results with a random classifier were poor, probably due to the class imbalance, so a random forest regressor was used with a custom threshold throughout the project.

Aim 1.3.1: Drop features with an overwhelming percentage of missing values and not enough correlation with the target variable

Using the training dataset (to avoid biasing the other datasets), missingness and correlation between each feature and the target variable (lethal outcome) were calculated. I dropped features with >20% missing values and <10% correlation with the target variable, or >60% missing values. While only the training dataset was used to calculate features to be dropped, the features were dropped from all of the datasets.

Aim 1.3.2: Utilize forward-filling for missing time-series data

I used the description document attached to the dataset to determine time-series variables. I then utilized a function which accessed the correct rows of the dataset and utilized built in pandas functions to forward-fill the missing columns.

Aim 1.3.3: Impute missing values

I tried two different imputation strategies. I created a copy of each dataset and created two groups: one with missing values filled in by single median imputation, and one with KNN imputation. I theorized that the KNN imputation would outperform the single median imputation, since missing values would be closer approximations of actual values. However, it was also possible that single-median imputation would outperform, since sometimes missingness in medical data is due to medical staff not reporting “normal” measurements. Essentially, if the data was generally missing at random, KNN imputation would outperform. However, if the data was missing not at random, I hoped this would be due to the “standard” values being unreported, which would be captured by single median imputation. In all cases, I fit the imputation on the dataset that would be used for training and transformed on the dataset that would be used for testing. I did this to avoid data leakage. For example, I fit on train and transformed on threshold, fit on train and transformed on validation, fit on tune and transformed on None, and fit on train + validation + threshold + tune and transformed on test (for final model training and testing).

Aim 1.4: Standardize continuous numerical features using SKLearn robust scaler

I found the continuous numerical features using the description document accompanying the dataset. I used the same strategy to prevent data leakage outlined above of fitting the scaler on training and transforming on testing. I used a robust scaler to avoid forcing the data to fit a normal distribution. I did this because in the context of patients undergoing a severe complication like myocardial infarction, the continuous measurements may not follow a normal distribution. I also had to gracefully handle the possibility of columns that needed to be standardized having already been dropped by previous steps, in order to make the code dynamic.

**Aim 2: Perform feature selection for target variable and for complication variables**

Aim 2.0: Define generic functions for feature selection

I wrote the functions to be used for feature selection and for displaying feature importance for a model. I wrote a function to fit a random forest regressor on the tune dataset, and then to call another function to graph the features by feature importance. I also created a function which could take in the ordered list of features and feature importance (returned by the scikitlearn random forest feature importance attribute), as well as the cutoff feature (“the elbow feature”), and return the features to be kept.

Aim 2.1: Feature selection for target variable.

I used the elbow method to select the cutoff feature (“the elbow point”) for the target variable for both single imputation (Figure 1) and KNN imputation (Figure 2), after building a model using a random forest regressor and plotting the ranking of the feature importance. I then utilized my generic functions to create and save the list of features to be used for each. All features above the selected cutoff were saved. This process is represented by the pink arrows in Figure 0.

Aim 2.2: Perform feature selection for each complication target variable on each imputation.

Using the same generic functions, I repeated the steps outlined in Aim 2.1 for each complication variable. The result was the optimal features chosen separately for each complication variable, and repeated for both the single imputation datasets and the KNN imputation datasets (Figure 3). This process is represented by the yellow arrows in Figure 0.

**Aim 3: Construct a machine learning random forest model for predicting lethal outcome in myocardial infarction patients, with the model having the ability to bias towards prioritizing recall**

Aim 3.1.0: Define generic functions

I created functions to be used by other functions in aim 3. Crucially, I created a custom weighted F1 scoring function, which is used for all scoring in aim 3.3 and on. This function operates on the same scale as F1, except that the weight of recall (and thus precision) can be customized. The customization is available at the top of the coding file, as the variable ‘recall\_factor’. For hospitals that want to prioritize minimizing false negatives, model testing can be performed using a recall\_factor higher than 0.5. This will allow hospitals to compare model performance against other models that they may have access to. I also created the final\_predictor function, which predicts and reports the score (using the scoring function specified by the user) given a training dataset, testing dataset, the features to be used as predictors, the target variable, and the threshold for the regressor to classify a case as positive (as optimized later on).

Aim 3.1.1: Calculate optimal threshold values for each complication variable and each target model

Using the threshold dataset, I found the optimal threshold value for each complication and target model. The target models tested included for both single and KNN imputation the following models: use all of the features and none of the complications, use the feature selection features and none of the complications, use all of the features and the complications, and use the feature selection features and the complications. I tested each possible threshold for each random forest regressor listed. The optimal thresholds were saved for each complication and each target model.

This step also provided key information related to maximum performance (maximum F1 score). The performance at the optimal threshold served as a ‘ceiling’ of performance that could be expected in testing. For example, the highest performing model with complications, with an F1 score of 0.706, was a ceiling above which a model attempting to predict complications could not surpass.

Aim 3.1.2: Perform “complication selection” (selecting which complications to use when predicting lethal outcome, and which to ignore)

Utilizing the results from 3.1.1, I performed complication selection. In other words, I dropped complications which did not have any strong performance across the threshold trials (<0.25 F1 score for all trials). To check my intuition that performance was stronger for complications with higher prevalence, I calculated prevalence for each complication. Indeed, the results showed a relationship between prevalence and performance (Figure 4). This indicates that poor performance on certain complications could be due to a low number of positive samples available for training. This process is represented by the green arrows in Figure 0.

Aim 3.2: Build models for predicting complication variables and report scores on validation data

For both KNN and single imputation, I built a random forest regressor model for each complication that was not dropped in the previous step. I used the features determined for the complication from aim 2.2, and the threshold determined for the complication from aim 3.1.1. I then reported the F1 scores for each. After comparing the scores across the KNN and single imputations, the KNN imputation F1 scores were overall higher (Figure 5), so the KNN imputation was chosen for the final choice of complication models to be used on the test data.

Aim 3.3: Build all variations of models and test on validation data, report results, and choose highest performers

Using the respective optimal threshold, I built each of the described target models in aim 3.1.1, training on the train dataset and testing on the validation dataset. This included testing models that used the actual complications results, which was just used for informational purposes. In real life, this model is impossible to build, since the complications are unknown. Weighted F1 scoring was used (note: in testing, recall\_factor was set to 0.5, resulting in regular F1 scoring being used). The highest performing model without using actual complications data or complications predictions, KNN all features, was saved for testing on test data. I then tested four models, each utilizing complication prediction. To utilize complication prediction means that the models took in the features, predicted the complications, and then used the combined features + complications vector to predict the target variable. (Technical side note: the raw regressor output, without use of a threshold, was used to improve performance.) I tested performance using feature selection and using all features for both KNN and single imputation. The highest performing model was KNN with all features, and it was saved for testing on the test data.

Aim 3.4: Test final models and report results, as well as report feature importance for interpretability

For each complication model, I reported the weighted F1 score on the test data (Figure 7, Table A) and displayed the feature importance of each feature for each complication model (Figure 6). I then trained the final two models (KNN all features without complications, and then with complications predictions) and reported the weighted F1 score, the precision, and the recall (Figure 9, Table B). I also displayed a graph showing the feature importance for each model (Figure 8). Finally, for a source of comparison, I built a baseline model which blindly used all features, was not scaled, and used single median imputation for missing values. I also used a random forest classifier rather than using a regressor with a custom threshold. I then reported results for the baseline model (Figure 9, Table B).

Significant Findings

I found it interesting that feature selection was unsuccessful in improving performance for the prediction of lethal outcome. For both the model that used complications prediction and the model that did not, using feature selection caused lower performance. I expected feature selection to improve performance, but I would be curious to see if a domain expert would have agreed with my hypothesis. The number of samples may have been too low, causing the feature selection to have too much noise to be meaningful. Alternatively, the features that were dropped might have each had a very small amount of information, which the model utilized to improve performance.

The performance on complication variables was quite poor. I hypothesize that this was due to the low number of samples available for the complications. I would be interested in speaking to a domain expert, because there are two additional issues that could be causing low performance. One would be if these conditions might go uncaught in some cases. In that hypothetical, the model is learning positive cases as negative, which would introduce noise into the training. Secondly, I wonder if these conditions are difficult to spot or if these diagnoses would be contentious. If diagnosing these conditions is a complex process upon which medical specialists may disagree, that would explain why these complications are more difficult to predict than lethal outcomes, for which there is no ambiguity between negative and positive cases. These are just two theories for why there might be too much noise in the labels for a model to correctly learn to predict the complications.

For comparison purposes, I tested the final models with recall importance set to 0.9. As expected, this resulted in the “complications + features” model outperforming the “only features” model (Figure 10). Note that all this does is affect the final formula of presenting recall and precision in one number: the recall and precision were identical in both cases, as expected.

It should be noted that KNN imputation outperformed single-median imputation, suggesting that the data may have been generally missing at random.

Finally, below are the relevant graphs and data:

**Figures:**

| Figure 0: overview of complete process. Blue circles represent features, red circles represent complications, and the green box represents lethal outcome. The yellow arrows represent feature selection for complication models, the pink arrows represent lethal outcome feature selection, and the green arrows represent complication selection for the lethal outcome model. |
| --- |
|  |

| Figure 1 from aim 2.1: random forest regression feature importance for lethal outcome - single imputation |
| --- |
|  |

| Figure 2 from aim 2.1: random forest regression feature importance for lethal outcome - KNN imputation |
| --- |
|  |

| Figure 3 from aim 2.2: Random forest regression feature importance for one of the thirteen complications (FIBR\_PREDS, or Atrial fibrillation). The graph shown is for single imputation- the process was repeated for each complication for KNN imputation. |
| --- |
|  |

| Figure 4 from aim 3.1.2: comparing % of samples with a given complication and the performance of a model predicting that complication |
| --- |
|  |

| Figure 5 from aim 3.2: complication model performance on validation data, single vs KNN |
| --- |
|  |

| Figure 6 from aim 3.4: random forest regression feature importance for the complication variable atrial fibrillation (‘FIBR\_PREDS’) |
| --- |
|  |

| Figure 7 from aim 3.4: complication model performance |
| --- |
|  |

| Figure 8 from aim 3.4: feature importance for the model using all features and complications predictions for KNN imputation |
| --- |
|  |

| Figure 9 from aim 3.4: comparing performance of final models on test data |
| --- |
|  |

| Figure 10 from significant findings: comparing performance of final models on test data, using recall\_factor = 0.9 |
| --- |
|  |

**Tables:**

| **Table A** from aim 3.4: complication model performance | Weighted F1 Score | Precision Score | Recall Score |
| --- | --- | --- | --- |
| Atrial fibrillation (FIBR\_PREDS) | 0.297 | 0.250 | 0.367 |
| Ventricular fibrillation (FIBR\_JELUD) | 0.079 | 0.044 | 0.375 |
| Pulmonary edema (OTEK\_LANC) | 0.273 | 0.375 | 0.214 |
| Chronic heart failure (ZSN) | 0.440 | 0.667 | 0.328 |
| Post-infarction angina (P\_IM\_STEN) | 0.197 | 0.158 | 0.261 |

| **Table B** from aim 3.4: final models on test data | Weighted F1 Score | Precision Score | Recall Score |
| --- | --- | --- | --- |
| KNN Model All Features | 0.565 | 0.545 | 0.585 |
| KNN Model Performance with Complications Prediction | 0.553 | 0.491 | 0.634 |
| Naive baseline | 0.245 | 0.750 | 0.146 |

Other Accomplishments

If a medical specialist specifically desires to predict certain complications, I reported the complication model results and displayed the feature importance graph for the complications. However, the F1, precision, and recall scores were not very strong for most of the complications.

# Changes/Problems

### Changes in Approach

I followed my original proposal plan through aim 2.1, and I coded the steps all the way through from my original plan (I left the code at the bottom of my Jupyter file). As described in the proposal, I used the tuning dataset to create a correlation matrix and I formed groups of correlated complications. For each group size two through five, each complication was placed in a group containing its optimal partners, such that the sums of the correlations between each of the complications in the group were maximized.

I then used the validation dataset to run multilayer perceptron models for each of the groups. One of my flaws at this step was using only the features from feature selection on the target variable. These features may have been the most relevant for the target variable, but it was a faulty assumption to think they would be the only relevant features.

I found the highest performing complication group for each complication, and then used this group to train on the training data and predict on the validation data. I augmented this complication dataframe with the feature selection dataframe, and used it to train another multilayer perceptron to predict the target variable. I did not flesh out this portion entirely - the preliminary results were so poor, I decided to abandon this direction and try something else. The performance with complications predictions (weighted F1 score of 0.40) was decidedly lower than the performance without complications (weighted F1 score of 0.46).

Another change to note is that I switched from the model type being a multilayer perceptron in my proposal to a random forest regressor in my final project. I realized that in my proposal I had overlooked the importance of interpretable results, and wanted to ensure medical specialists could interpret model results. Since random forests can show feature importance, I switched to random forests.

### Problems or Delays Experienced and Corrective Actions Taken

Outside of the usual programming workflow of bugs needing to be sorted through, I did not encounter significant delays that required severe corrective action.

# Impact

### Impact on the Domain

I hope that my project can lead to improved plans of care, which can lead to improved outcomes for patients who experienced a myocardial infarction. Providing medical specialists with an interpretable tool for predicting lethal outcomes can assist them as they advise and monitor their patients. Hospitals can customize the recall importance when testing the model, which can help them balance the importance of avoiding false positives (precision) and false negatives (recall) when understanding model performance. Ultimately, patients can live longer and healthier lives.

### Impact on the Individual

I took this class to learn how to apply machine-learning techniques to real-life datasets. This project gave me the opportunity to solve a machine-learning task starting from scratch - with just a dataset. I was able to experience what it is like planning and proposing a project, which immensely improved my critical thinking skills. I became a better data scientist by applying the best practices we had discussed in class, as well as having to come up with my own solutions to the problems that I encountered. Ultimately, I improved my programming skills and the depth of my machine learning understanding by handling a real-life case as a data scientist. I am very thankful to have had this experience, and I feel more confident tackling machine-learning tasks in the future.