MSDS 630: Advanced Machine Learning Final Project Report

Da Sea Mate

Daren Ma
(kaggle@Daren Ma)
[User ID 4435709]

Sean Tey
(kaggle @seantey)
[User ID 1003859]

Table of Contents:

Abstract:

The goal of this project is to predict the mean Heart Rate (y_mean_HR) and mean Mean Arterial Pressure (y_mean_MAP) simultaneously based on the given data set, where the average R-square is our North Star metric. We first understood the patterns of the data through plotting the maximum, average, and minimum values of the features we care about and filtered out some outliers found in the EDA stage.

As part of the feature engineering step, we exploited the time-series like property of the numerical features within a key in conjunction to borrowing some ideas from Topological Data Analysis and we created new features based on a rolling average with different window widths. We performed binning to age because of its high cardinality, and preserved the mode values of other categorical features. We splitted the training and validation set based on different patient ids to prevent mixing data from the same patients in both the training and validation sets thus preventing data leakage.

After experimenting with various machine learning algorithms, we picked CatBoostRegressor given its great performance and GPU integration which resulted in faster training time. Grid Search was applied to find the optimized hyperparameters. Up to now, our mean validation R-square between the two targets is 0.918. In the next step, we plan to focus more on the prediction of MAP and implement more feature engineering techniques.

Dataset Description:

The training dataset has 1348470 rows and 17 columns, among which the targets are the key-wise mean HR(Heart Rate) and MAP (Mean Arterial Pressure). Among the features, there are three groups of data. Age and Gender are self-evident and are categorical features. Then we have unknown categorical features x1 to x6 and unknown numerical time-series features xx1 to xx5.

The data is organized into two layers: the patient_id and the key. In total there are 2321 distinct patients and 44949 keys. For each patient, there can be one or multiple keys. For each key, there is always 30 rows of data based on our observation. However, we need to keep in mind that the values in the same key may still vary a lot or behave identically and hopefully the model or further feature engineering will capture these patterns.

Exploratory Data Analysis:

We implemented the EDA in the following steps and found several patterns to help our feature engineering.

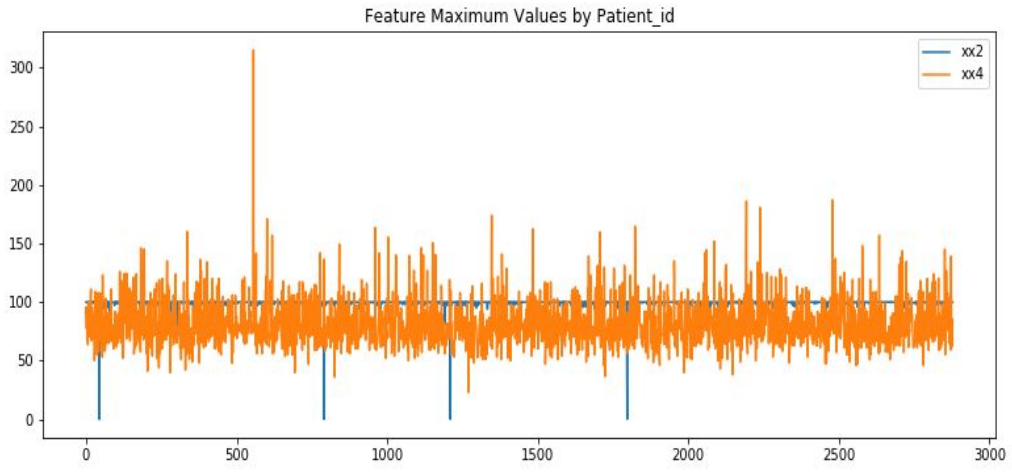
1. First we did the regular DataFrame.info() and DataFrame.describe() to look at the basic feature data types and statistics.

   We found out that there is no Null value in this dataset, freeing us from worrying about missing value imputation. Also, from the dtype of the features, we were able to tell that the features age, gender, x1 to x6 are all categorical, while xx1 to xx5 are numerical values.

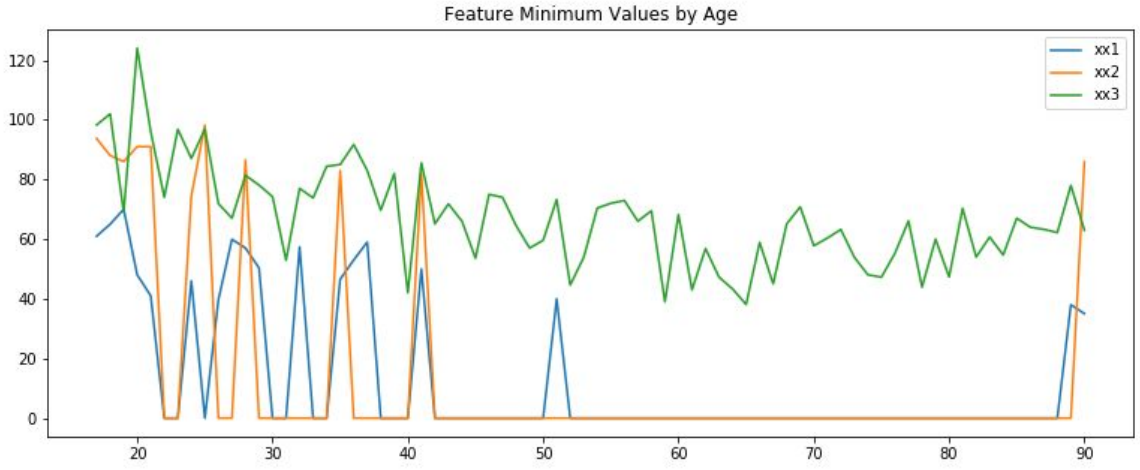
2. Inspired by some basic medical common sense, we plotted the training data by age, and tried different combinations of features to investigate the patterns in the data in terms of age.
   Fortunately, in this process we found two important patterns:
   1. Feature xx1 and target y_mean_HR have high collinearity. In later experiments, even predicting HR with barely xx1 gave us a decent R-square higher than 0.9.

   2. Feature xx5 and xx6 have high collinearity. They look like the boundaries of some kind of medical signal such as blood pressure.

3. Further investigation led us to look at the maximum and minimum values of the features.
   1. The max value of xx4 definitely has an outlier, and the max value of xx2 among all the patients is either fixed 100 or 0. This is probably some kind of anomaly happening to the patient or the data collection.



2. Based on the following image, we found that the minimum values of xx1,xx2, and xx3 also have outliers or at least strange values. In detail, there are so many zero values in xx2 and xx3, bringing concern that these features should be taken care of in the feature engineering part.



Feature Engineering:

Every patient id is associated with several keys, some patients have more keys than others and each key has exactly 30 data points associated with it. Since all of those data points that share the same key also share the exact same target values for y_mean_MAP and y_mean_HR, we will also aggregate the data based on the keys.

1. For all continuous variable columns xx1, xx2, xx3, xx4, and xx5, we aggregate by Key then generate simple summary statistic and add columns such as
    a. mean
    b. median
    c. min
    d. max
    e. standard deviation
    f. interquartile range

    This generates 5 x 6 = 30 new columns.

2. Since all discrete variables are the same within the 30 data point window for a given key, we just take the mode to preserve the original value when aggregating by key.
    a. Preserve values for x1, x2, x3, x4, x5.
    b. Note that taking the mode may require unnecessary additional computation but it is easier to use a default pandas function to do the aggregation so the additional computation time is offset by the convenience.

3. Generate a new column called age_group using the discrete value age column with the following rules:
    a. Assign value 0 to age [0,10]
    b. Assign value 1 to age [11,20]
    c. Assign value 2 to age [21,30]
    d. Assign value 3 to age [31,40]
    e. Assign value 4 to age [41,50]
    f. Assign value 5 to age [51,60]
    g. Assign value 6 to age [61,70]
    h. Assign value 7 to age [70,infinity]

4. Borrowing ideas inspired from topological data analysis
    a. Within the 30 time-series data points for a given key, we take a rolling average window of new points and use each of those points as a new feature.
    b. This is like going from long to wide format for every 30 data points of the key
    c. Depending on the window size, the additional “dimensions” added will shrink. For example, a rolling window of size 3 on 30 data points means we will be left with 28 data points.
    d. A window of size 3 generates 28 new columns for each of the columns x1, x2, x3, x4, and x5. Which results in 5 * 28 = 140 new columns
    e. We can also repeat this with window of sizes 2, 5, 10, 20, 25

Machine Learning Methods:

We applied various popular machine learning models and compared their performance on this dataset. To be specific, we used Scikit-learn's RandomForest, XGBoost, and the CatBoost packages in Python and found that the gradient boosting algorithm worked best so far. Our current choice for modeling HR and MAP is to use the CatBoostRegressor (Gradient Boosted Tree) for both.

Experimental Results:

We tested various feature engineering methods starting with generating summary statistics columns which gave us a R2 score of around 0.875. After that we added the moving average window of size 3 as new columns and that increased our score significantly to 0.914. Then we added additional different moving average window columns of size 2,5,10,20,25 and our score went up slightly to 0.917.

We also conducted hyperparameter tuning of the model using CatBoost's grid search tool with the following list of hyperparameters:

- Depth: [2,3,4,5,6,8]
- Iterations: [1000, 1500, 2000, 2500, 3000, 5000, 10000]
- Learning Rate: [0.01, 0.05,0.1, 0.5, 1]
- L2-Leaf-Regularization : [1,2,3,5,7,9]

We found the following set of hyperparameters to work best at the moment:

- Depth: 4
- Iterations: 5000
- Learning Rate: 0.1
- L2-Leaf-Regularization : 2

With this we were able to increase our score to around 0.918 as of February 22nd.

List the responsibilities:

List the responsibilities of each person in the group:

- Daren Ma:
- EDA
- EDA notebook
- Tried: Random Forest, XGBoost, MultiOutputRegressor
- Write report

Sean Tey:

- EDA
- Pre-processing notebook
- Tried: Regularized Regression /w ElasticNet, GBT /w CatBoostRegressor
- Write report

Repository Link:

All of the code used in this report can be found at:
https://github.com/USF-ML2/final-project-da_sea_mate