Machine Learning Engineer Nanodegree

Capstone Project

Edy Zhang

Dec 27th, 2017

Definition

Project Overview

Delivering babies is not only stressful for the family, but also intense for the supporting doctors and nurses. Different than most other treatments and operations, parturition can take place at any time, which makes OBGYN a 24-7 department in most places. Generally, doctors look at the fetus heart rate plot to determine the health status of the fetus, namely putting them into three categories (1 to 3); 1 being the healthy status, 3 being the unhealthy status that needs immediate attention and/or a C-section, 2 being the grey area that requires more attention. The unattended category 3 fetus can suffer brain damage or even death in a matter of few minutes. The widen usage of Pitocin, which intensify uterus contraction, further expedite the damage. With the mix of different level of experience, fatigue due to long hours and intense work load, and short of staff at late hours, misclassification of fetus status and slow response to category 3 patients have been causing numerous incidents in an ongoing manner. This not only result in broken heart families, but also cost billions of dollars in lawsuit to the industry. Similar studies have been done, such as *A Machine Learning Approach to the Detection of Fetal Hypoxia during Labor and Delivery* [1]; however, it achieved only 50% recall with false positive rate at 7.5%. With the rise in machine-learning technologies, I see the opportunity to create a product that dynamically classifies fetus health status that acts as a second eye for doctors, in the goal of prevent tragedies.

In this project, I created a publicly available API, which is capable of predicting fetus health status based on a list of inputs. The API uses a classifier trained using UCI Machine Learning Repository Cardiotocography Dataset [2] and AWS as supporting infrastructure.

Problem Statement

The goal is to create a publicly available API running on AWS capable of predicting fetus health status based on a list of inputs; the tasks involved are the following:

1. Download and explore the UCI Machine Learning Repository Cardiotocography Dataset to understand the data
2. Preprocess to clean up the dataset and prep for model training
3. Compare various classification algorithms and their performance on this dataset
4. Train and fine tune the selected algorithm, and export the final model
5. Create API that takes necessary input parameters, uses exported model for prediction, and returns results in appropriate format
6. Deploy created API to AWS, and use respective AWS technologies to make it publicly available at a reasonable cost

Metrics

As the dataset is skewed towards fetus with category 1, accuracy will not be a good evaluation metrics. Instead, Fbeta-score is a great option since it puts customizable weight on both the false positive and false negative.

In addition, the accurate classification of category 3 fetus is intuitively much more important than category 1 or 2, because any miss-catch on category 3 fetus could highly possibly result in injury. As a result, at Fbeta-score evaluation, both category 1 and 2 will treated as negative, and category 3 will be treated as positive. False positive, false negative, recall, precision, and Fbeta score, in the case, can be interpreted as:

- False Positive, fetus with category 1 or 2 categorized as category 3
- False Negative, fetus with category 3 categorized as category 1 or 2
- Recall = TP / (TP + FN), out of all category 3 fetus, what percentage was predicted as category 3; recall tells what percentage of category 3 fetus can be caught and potentially saved

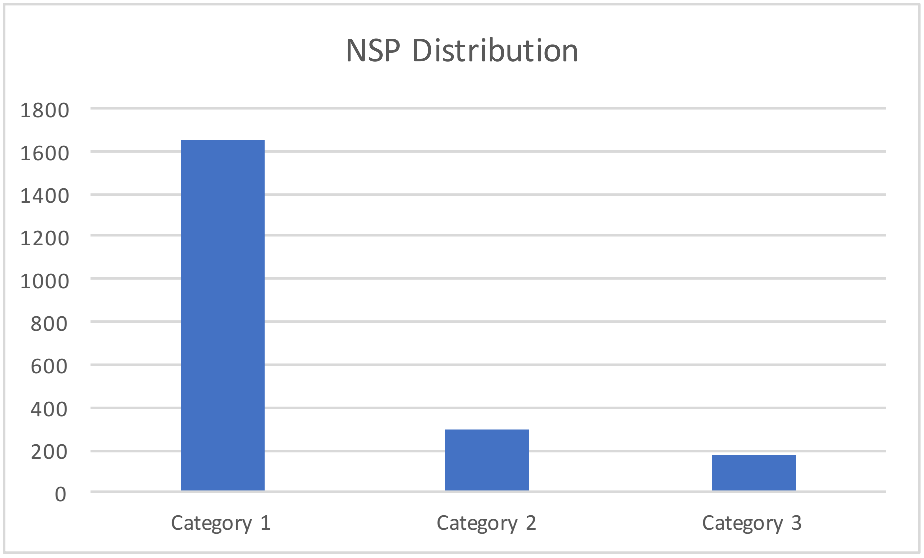
- Precision = TP / (TP + FP), out of all predicted category 3 fetus, what percentage was truly category 3; precision tells what percentage of medical staffs' efforts gears towards a real medical complication
- Fbeta = (1 + Beta^2) * Precision * Recall / (Beta^2 * Precision + Recall); Fbeta is the balance between medical staffs' efforts and category 3 fetus being saved

Ideally, a model that achieves both a high recall and precision would be optimal; however, when only one can be optimized, clearly a higher recall is more valuable than precision. As illustrated by the above equation, a Fbeta score with beta larger than 1 shall be used.

Analysis

Data Exploration and Visualization

The UCI Machine Learning Repository Cardiotocography Dataset contains 2126 instances, each with 40 attributes including the NSP output attributes (3 category class codes). As the inputs are readily available attributes from most fetus heartrate monitors, a model created based on this dataset can potentially be a plug-in for existing heartrate monitoring systems.



Out of the 2126 data points, 1657 having category 1, 297 having category 2, and 178 having category 3. Like most disease related dataset, this dataset is skewed towards the healthy category. Example row:

FileName	**Date**	**SegFile**	**b**	**e**	**LBE**	**LB**	**AC**
Variab10.txt	12/1/96	CTG0001.txt	240	357	120	120	0
FM	**UC**	**ASTV**	**MSTV**	**ALTV**	**MLTV**	**DL**	**DS**
0	0	73	0.5	43	2.4	0	0
DP	**DR**	**Width**	**Min**	**Max**	**Nmax**	**Nzeros**	**Mode**
0	0	64	62	126	2	0	120
Mean	**Median**	**Variance**	**Tendency**	**A**	**B**	**C**	**D**
137	121	73	1	0	0	0	0
E	**AD**	**DE**	**LD**	**FS**	**SUSP**	**CLASS**	**NSP**
0	0	0	0	1	0	9	2

Out of the 40 attributes, some are noticeably removable, such as FileName, Date, SegFile (file name related), b and e (time related), and a 10-class classifier that's one hot encoded (column A, B, C, D, E, AD, DE, LD, FS, SUSP, CLASS). Below is the description for the rest attributes:

- LBE: heartrate baseline value from medical expert
- LB: heartrate baseline value from SisPorto
- AC: heartrate accelerations
- FM: fetal movement
- UC: uterine contractions
- ASTV: percentage of time with abnormal short term variability
- MSTV: mean value of short term variability
- ALTV: percentage of time with abnormal long term variability
- MLTV: mean value of long term variability
- DL: heartrate light decelerations
- DS: heartrate severe decelerations
- DP: heartrate prolongued decelerations
- DR: heartrate repetitive decelerations
- Width: heartrate histogram width
- Min: low frequency of the heartrate histogram

- Max high frequency of the heartrate histogram
- Nmax: number of heartrate histogram peaks
- Nzeros: number of heartrate histogram zeros
- Mode: heartrate histogram mode
- Mean: heartrate histogram mean
- Median: heartrate histogram median
- Variance: heartrate histogram variance
- Tendency: heartrate histogram tendency, left assymetric, symmetric, right assymetric
- NSP: normal=1, suspect=2, pathologic=3

Conveniently, all of the above attributes are in number format, integer, float, or binary. As suggested by the descriptions, some of the attributes could be correlated and eliminable.

Algorithms and Techniques

Multiple classification algorithms are tried and performance compared; these includes Gaussian Naïve Bayes, Logistic Regression, KNN, Decision Tree, Random Forest, and Neural Network. Gaussian Naïve Bayes and Logistic Regression are selected based on their ability to work well with small dataset. KNN, although not performing well on high dimensional and small dataset, is selected on it's fast training speed and used for comparison. Decision Tree is robust to outlier, and famous for various practical use cases. Random forest, inherit the strengths from decision tree, is less prone to overfitting. Last but not least, neural network is selected for its nowadays popularity and outstanding industrial performance.

The final selected algorithm, Random Forest, is then fine tuned on the following features to achieve an even higher Fbeta score:

- max_features using algorithm sqrt and log2
- max_depth from 3 to 50
- min_samples_split from 2 to 5

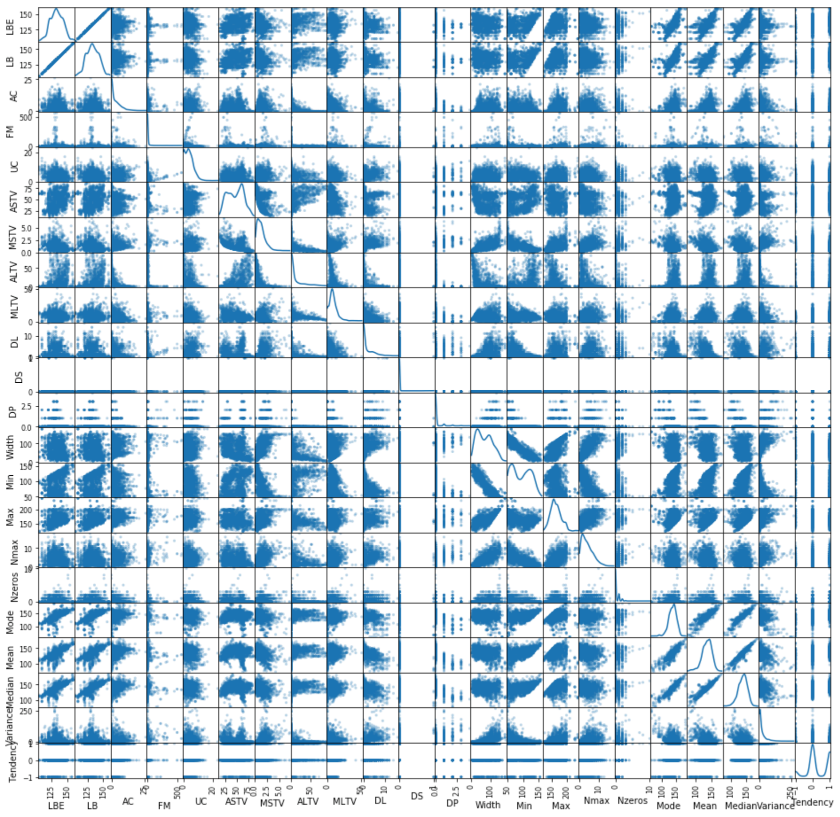
Benchmark

Based on radiopaedia.org, a health fetus will have heartrate between 120 and 160 bpm. I will use a hard threshold, 120 – 160 bpm, to simulate a traditional fetus health predictor, and use it as the benchmark model. This benchmark model simply predicts a category 1 when heart rate is between 120 – 160 bpm, and category 3 when heart rate is out of this range. A Fbeta score is calculated against this simple model for comparison and performance evaluation. With a beta of 3, the fbeta score is 0.59 and recall is 0.62.

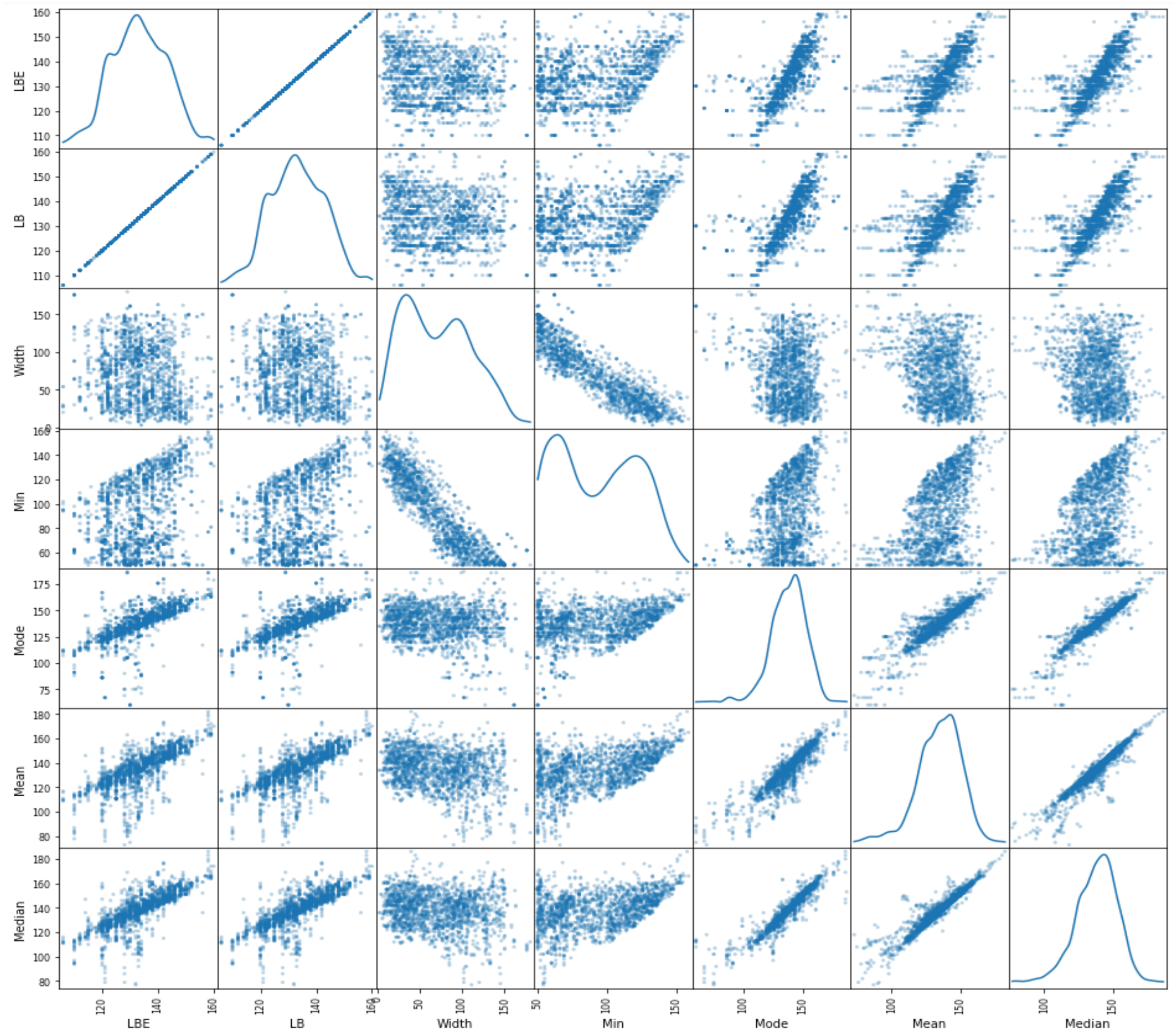
Methodology

Data Preprocessing

Given there're only around 2,000 data points, while each with 40 attributes, the curse of dimensionality will cause poor performance on most algorithms. Therefore, the foremost cleanup to do is to reduce the number of irrelevant attributes as much as possible. As described in the Data Exploration and Visualization section, based on the column description, 11 columns are 10-class classifier related and removed, 2 columns are filename related and removed, and 3 columns are date-time related and removed. Furthermore, column DR has the same value for every data point, which is irrelevant for modeling, and therefore removed. After separating NSP (outcome) column from the input attributes, the number of attributes is down to 18 at this point. A scatter matrix is then plotted among the remaining input attributes in the goal of



discovering and removing highly correlated columns (graph above). A second scatter plot is then drawn regarding the suspected highly correlated columns; this include attribute LBE, LB, Width,

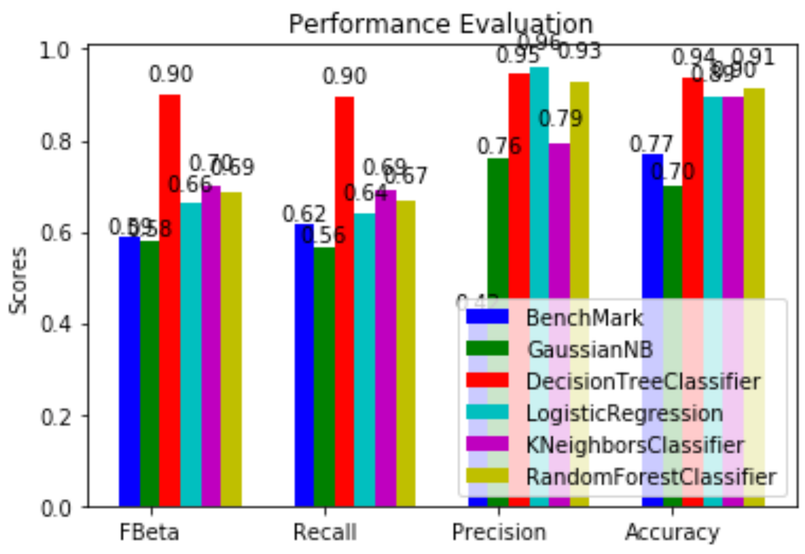


Min, Mode, Mean, and Median (refer to the following graph). According to the plot, LB and LBE are highly correlated, min and width are correlated, and the group of mode, mean, and median are correlated. To reduce dimensionality, only one in each group shall be left for model building. LB is selected based on the description, SisPorto, so that the selected attributes (AC, FM, UC, ASTV, mSTV, ALTV, mLTV) are uniformly from SisPorto. Wdith and Median are selected because they are more uniformly distributed. With this step, 18 attributes are left as input attributes for model creation.

The next step is to identify the columns with highly skewed distributions and possibly extreme values, and transform such columns so that the extreme values wouldn't have overpowering effect on the model. Based on the diagonal in the first scatter plot (data distribution of each attribute), the following columns are selected to go through log transformation: AC, FM, UC, MSTV, ALTV, MLTV, DL, Nzeros, Variance. Finally, a min-max-scaler from sklearn is applied to all attributes to bring all values to range 0 to 1.

Model Implementation

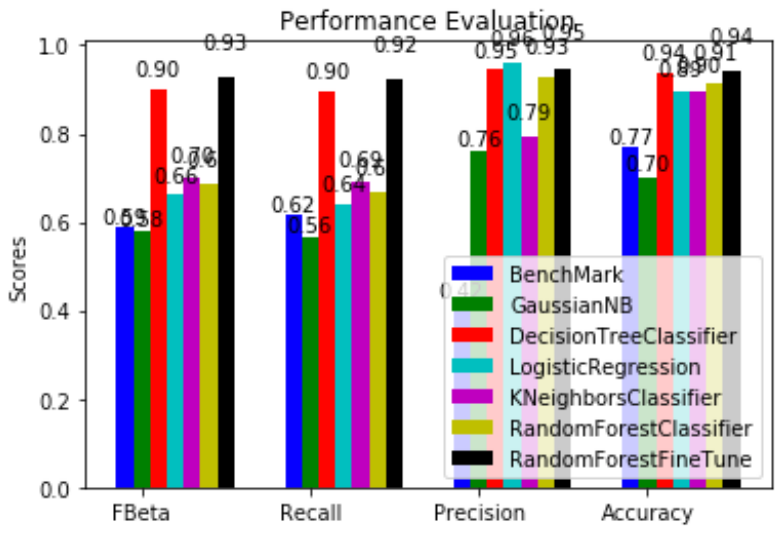
Five classification algorithms from sklearn were tried and performance compared; these includes Gaussian Naïve Bayes, Logistic Regression, KNN, Simple Decision Tree, and Random Forest. According to performance evaluation, decision tree stands out among the five, having recall and fbeta around 90%, and 30% better than the second best performing model.



Undoubtedly, it is selected for fine tuning. In addition, Random Forest is also selected for fine tuning phase not only because it's in the same family as decision tree, but also theoretically, it is more robust and less prone to overfitting than simple decision tree.

Model Refinement

In fine tuning phase, grid search method was applied to both Decision Tree and Random Forest, and a customized scorer was created to evaluate the models focusing on category 3 fetus. Interestingly, with countless tuning and trials using various parameters, such as max_features, max_depth, min_sample_split, etc, the performance for simple decision tree were never able to improve beyond the un-tuned model. And as expected, the fine tuned Random Forest achieved



a performance better than simple decision tree: Fbeta reaches 92.5% and recall is 92.3%. This Random Forest model is then exported using Pickle for API usage.

Neural Network Model Creation and Challenges

A multi-layer neural network model was created using Keras. One of the first challenges was to correctly and effectively train the model on this highly skewed data. With categorical crossentropy as loss function, the model improves its accuracy rapidly while keeping the recall at a low value. With this observation, I duplicated the data for category 2 and 3 in the goal to make it uniformly distributed and therefore effectively trained using this network. Based on the ratio among category 1, 2, and 3 in training set, category 2 data was copied five times, and category 3 nine times, in order to roughly match the number of category 1 data. The training set is then shuffled before use. With this trick, the recall and fbeta quickly reached above 80% for both training and testing set.

The second challenge was, possible with the data duplication trick, the mode is easily overfit. The final model has 18 input attributes, 18 nodes on first layer, and 3 nodes on second layer as output; both layers are fully connected layers. With countless attempts, any additional layer would introduce overfitting, and any more nodes on the first layer would also introduce overfitting. With such architecture, the model is able to achieve an acceptable performance, at 85% for fbeta and 87% for recall. However, with the unbeatable performance by Random Forest, the model of choice is still Random Forest.

API Implementation

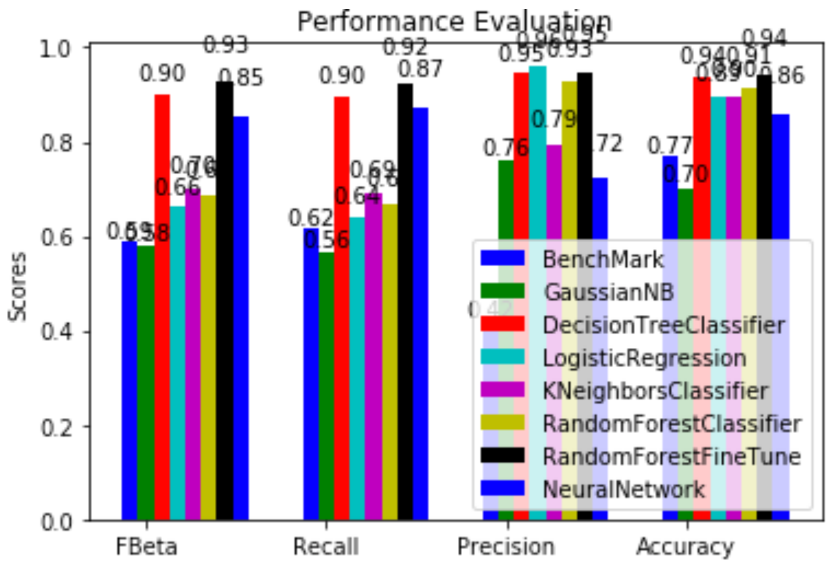
A python application is created in the goal of running in AWS as a REST API. It uses flask[3] to handle REST requests and responses; POST request body is expected to be in json format and contains all 18 attributes used in the classifier. The input attributes are cleaned in the same fashion, log transform and min-max-scaled, as the data used for model training and testing. The models are stored on AWS S3, and retrieved by application using boto3[4] library. After prediction, the result is send back to client in json format.

The python application is deployed using Zappa[5] to AWS Lambda; an AWS API Gateway was also created by Zappa as the front gate to handle requests to this Lambda. This API has been made publicly available and has the potential for systematic integration and real life usage. Please refer to the readme file for API usage details.

Results

Model Evaluation, Validation, and Justification

As briefly discussed in fine tuning section, the fine tuned Random Forest has the best performance among all in terms of the customized Fbeta score and recall. For testing set, this



model reaches a Fbeta of 92.54%, recall of 92.31%, precesion of 94.74%, and accuracy of 94.13%. Out of the 426 samples, it caught 36 true positive category 3 fetus, with 3 false negative misses. Comparing to the benchmark model with less than 60% Fbeta score, the resulting model is model than acceptable.

The performance of the neural network model, although disappointing, is somewhat expected. The lack of data, especially with such a skewed dataset, is one of the leading contributor. Knowing neural network tend to perform well with vast amount of data, a better model with neural network is still achievable once more and more data are made available in this domain.

Conclusion

Reflection

The end resulting product from this project meets me expectation, although I was imagining a model with 100% recall and lower precision. The data cleaning and tuning for neural network has been the most cumbersome sections given the limited amount of data.

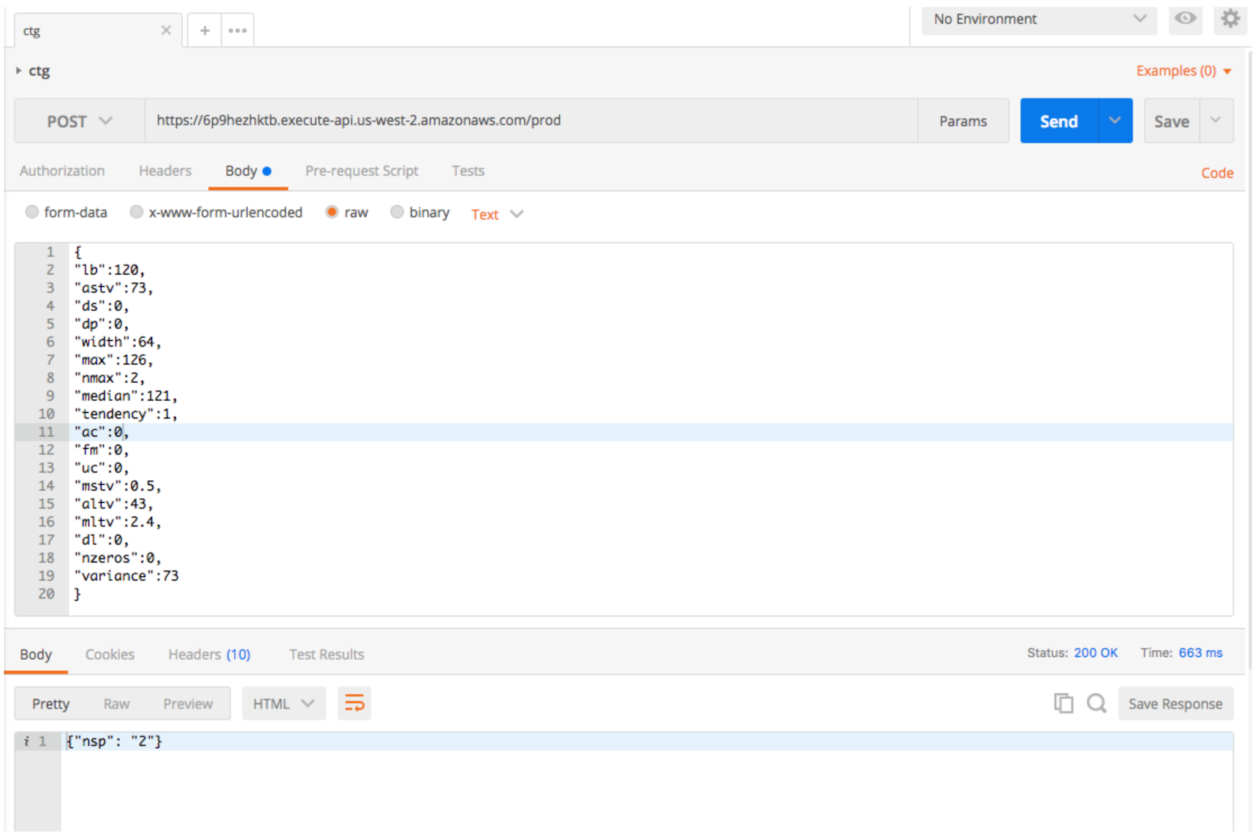
The AWS Lambda setup was also a great learning experience; unexpectedly, running sklearn models on AWS Lambda is not easy to setup given its code size limitation. As a workaround, the code has to be deployed to S3, and retrieve from S3 every time it needs to run, which made the application suffered in response time. However, given the scope of the project and the financial cost and maintenance concern, AWS Lambda might still be the best option to setup such a simple API and make public.

Improvement

Same as many machine learning obstacles, a better or even productionalized fetus health status model need a lot more data. Such productionalized model shall also investigate the possibility of tapping into existing fetal heartrate systems for integration and data retrieval. To create a universal usable model, the project shall compare the different parameters available is various systems, and select the common and usable attributes as input. The possible collaborations with existing fetus heartrate system vendors is the best way to success.

Demo of App and Visualization

To use this API, please make a POST request to the following url: https://6p9hezhktb.execute-api.us-west-2.amazonaws.com/prod. The request body shall include all 18 attributes in json format, example shown below. The response will be a prediction ranging from 1 to 3; 1 being health, 2 being suspected, 3 being pathological.



POST https://6p9hezhktb.execute-api.us-west-2.amazonaws.com/prod

Body: raw, Text

```
{
"lb":120,
"astv":73,
"ds":0,
"dp":0,
"width":64,
"max":126,
"nmax":2,
"median":121,
"tendency":1,
"ac":0,
"fm":0,
"uc":0,
"mstv":0.5,
"altv":43,
"mltv":2.4,
"dl":0,
"nzeros":0,
"variance":73
}
```

Status: 200 OK Time: 663 ms

```
{"nsp": "2"}
```

References

[1] *A Machine Learning Approach to the Detection of Fetal Hypoxia during Labor and Delivery*, by Philip A. Warrick, Emily F. Hamilton, Robert E. Kearney and Doina Precup https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwjL47v2gYjYAhVr2oMKHdGOBAQQFggsMAA&url=https%3A%2F%2Fwww.aaai.org%2Focs%2Findex.php%2FIAAI%2FIAAI10%2Fpaper%2Fdownload%2F1597%2F2372&usg=AOvVaw07ZNZJvDTi6NyI8BUqenlm

[2] https://archive.ics.uci.edu/ml/datasets/Cardiotocography#

[3] http://flask.pocoo.org/

[4] https://boto3.readthedocs.io/en/latest/

[5] https://github.com/Miserlou/Zappa