In Woo Park

ICS 635 Machine Learning

13 February 2020

Kaggle Competition

**1. Specify features and pre-processing**

My submission for the Kaggle competition was heavily influenced by the example submission provided by the Professor. I followed the format of loading training data, loading the test data, combining the data with patient data (i.e. ICU), and finally making a prediction using logistic regression. When loading the test data, I used one hot encoding for the state variable. The other relevant features that I investigated were age\_group, icu\_yn, and death\_yn. I thought these were the best choices to analyze first as they are associated with COVID more than the other features in terms of relevancy.

The first thing I noticed was that the training data set had numerous missing values for relevant attributes. I attempted to handle the missing values by depending on a correlation between attributes. For instance, if the ‘death\_yn’ attribute had a value of either 0 || 1, regardless of the value, we can fill in relevant attributes using this information. For ‘death\_yn’ to be reported, it must be in a medical facility. Therefore, I attempted to fill in data in the small training set by: if ‘death\_yn’ had a value of 1, fill the attributes ‘icu\_yn’ as 1, ‘hosp\_yn’ as 1, ‘symptomatic\_yn’ as 1, ‘labconfirmed’ as 1, and ‘exposure\_yn’ as 1. This method would be overfitting values but it only would apply to missing values. If ‘death\_yn’ had a value 0, I would fill the attributes the same way except ‘icu\_yn’ as assuming they survived the ‘icu\_yn’ without any additional information that confirms this hypothesis would rely on a, “hunch” and that would be REALLY overfitting values.

**2. Specify data splits and how they are used**

The train\_small dataset had a size of 100,001 which is what I used as the dominant training data. I am not sure how the example submission permutes or divides the training set, validation set, and test set. I will be looking into this because I do not like the fact that I do not understand how this works. As for now, I did not modify the data splits in any way for my particular submission.

**3. Specify the hyperparameter search space**

As explained in **[2]**, nothing was changed about data splits from the example submission and in addition, nothing was changed about the hyperparameter search space. Therefore all range values were explored.

**4. Explain how hyperparameters were optimized**

After trying all combinations of hyperparameters in the search space, the model with the highest accuracy on the validation set was selected.

**5. Evaluate model on clean test set**

All submissions to Kaggle were based on a clean test set. I pushed 5 total submissions to Kaggle with the highest prediction rate at 0.93448 (private) and 0.93602 (public). In this submission, I specified the data type of ‘death\_yn’ as a float and made the program load the test data test.csv with ‘age\_group’ as the relevant feature to be one hot encoded. I chose ‘age\_group’ as it’s intuitive that ‘age\_group’ is a significant indicator when predicting COVID deaths according to the CDC. This method provided the highest personal record.

My other attempts were trying to concatenate multiple attributes, attempting to depend on only ‘age\_group’ and ‘underlying\_conditions’, and one hot encoding other random attributes to see what happens. None of these attempts were as successful as my explanation above.

**6. Explain any differences in the train/test datasets**

From what I’ve noticed, the test set has a range from September 30, 2021 through November 30, 2021, the full training set has a range from December 31, 2019 to August 31, 2021, and the small training set only reports data from August 31, 2021. Large data dumps that are related but that aren’t sharing similar date ranges could harm model performance. In addition, only the smaller training set had the ‘death\_yn’ attribute which probably carries the majority of the prediction but once we move to the full training set we lose that attribute. Both training sets also have numerous empty values which was up to me in how I wanted to handle the missing data.

**7. Additional Information**

My steps for this assignment was to: [1] look into one hot encoding, [2] take the training set and handle the missing values, [3] follow the models presented in the class lectures and [4] apply a decision tree model into the cleaned data set.

I got stuck on [3] as I couldn’t get the syntax to work for decision trees. This is no one's fault but mine as I am not familiar with python and it’s syntax for list comprehension and additionally ensemble and decision tree syntax. Although I followed the recorded lectures I just couldn’t get it to work. I assume that changing the one hot encode to ‘age\_group’ and applying one of the models (i.e. ensemble, decision tree) would have provided me with a more accurate score and subsequently I would not have had to write this analysis document on the Kaggle competition.