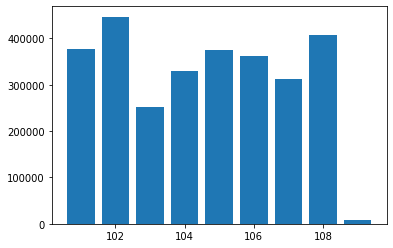
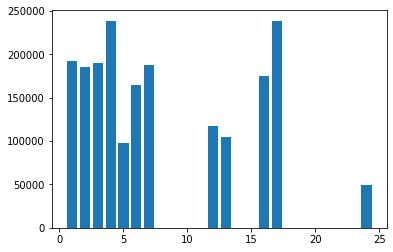
For this assignment, we are given the task of recreating the analysis done by Chowdhury et al. (2017) on the classification of physical activities through ensemble methods. The data in question covers nine individuals who wore special bands (called IMUs) to track their physical data. The goal of the study was to investigate how both sedentary and physical activity are related throughout the day, as well as to investigate whether the use of ensemble learning algorithms improve physical activity recognition accuracy compared to the single classifiers. The ensemble method is carried out in the corresponding Jupyter notebook.

Data on 54 features is collected, with many of these directly related to each other. For example, the positions and temperature of the hand, ankle, and chest were collected throughout the day but each is listed as its own column. There are timestamps, heart rates, activity ID’s and data from the IMUs. These bands contained information on temperature, acceleration, gyroscope and magnetometers. The Activity ID is treated as the classification target in this problem, with the 24 classes being the 24 activities that were prescribed. The nine individuals included one woman and eight men, with average age of about 27 and BMI of 25.

Before we can train our classifiers, we must perform some preprocessing. We start by defining a map for the target, then define some functions to generate comprehensive columns for the IMU data. We then load all 9 subjects in (through a for loop) and combine them into one master set. We first notice that we have many missing values, and we also see that each subject therefore has differing amounts of information (with subject #9 in particular having lost a lot of information) and this is shown in the frequency plot below:



Additionally, each data point is collected sequentially, and thus we will need to treat the data as though it were a time series model. Finally, we pay attention to our target variable. This is a categorical output and has 24 different levels. These categories represent different activities that were prescribed by the researchers. They range from sedentary motions, such as sitting or lying down, to everyday household movements like vacuuming or driving, and even more rigorous sporting activities like playing soccer and jumping rope. It is important we know the frequency of this variable, so the chart below demonstrates this.



Before plotting, we eliminated the value for 0, as it represents transience (switching between activities). A few values are outright missing, as the researchers did not define a class for them (these are 8,14,15,21,22,23). We notice a few activities, like computer work and driving, were not counted at all, and others like walking and ironing were accumulated the most. While certainly peculiar, we can possibly attribute this to the test subjects already living at home where they would be performing activities like vacuuming and walking up/down stairs as opposed to driving and working at a computer desk.

The paper by Chowdhury et al. addresses the NaN problem through linear interpolation, so we will do the same. We also follow their step of treating the data as time series, and we drop all values with the null (undefined in the map) activity. After conversion, and splitting into X,y dataframes, we are left with an input matrix of size 1,942,872 x 54. Because of the nature of the time series data and the fact that we technically have nine separate data matrices (one for each subject), we will take advantage of this when deciding how to split into training and testing sets. Rather than taking a random sample from the entire master set, we select two arbitrary subjects (which turned out to be #4 and #7) and let them be the testing subset, and everything else is the training set.

Now recall the theoretical definition of the ensemble. In a Q-classifier ensemble, let the classifier set and set of classes be defined as  respectively. Each classifier  produces a class label  without any further information. When classifying an object , the Q classifier outputs a vector  Then, the decision fusion techniques combine the classifier’s output and provide a single class label. Put more simply, each classifier “votes” to assign a record to a particular class. The final decision can be made several ways, including averaging the results, hard voting (sort of like paper ballots) or stacking, which generates a new dataset using inputted models that can be used by the ensemble.

For our ensemble, we implement the same three classifiers as from the article: Random Forest, Boosting, and Bagging. Random Forest works by building *n* decision trees using a small subset of the data, then utilizing its own voting procedure (based on each tree that was built) to assign records to classes. Boosting is a general term for many classifiers, but we choose to use the XGBoost algorithm in this case. This works by implementing gradient boosted decision trees in an additive model in a forward stage-wise fashion and thus allows for the optimization of arbitrary differentiable loss functions. Similarly, bagging is a general term with many possible specific models, but the BaggingClassifier function from the sklearn ensemble package is utilized here. A Bagging classifier fits base classifiers each on random subsets of the original dataset and then aggregates their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance a decision tree by introducing randomization into its construction procedure and then making an ensemble out of it. Note that in this case, samples are drawn with replacement, so it firmly is defined as bagging (as opposed to pasting).

Now that we implement the classifiers and the resulting ensemble, we can interpret the results. Recall that the training and testing data were split by choosing a random two subjects to compare against the other seven. Also note that this is an enormous dataset (almost 2 million records in 55 columns) and it requires some time to process. What we notice is a likely problem with overfitting. This is because the score metric in the sklearn library looks for mean accuracy of the predict algorithm in ensemble package. Here, it is returning values for almost or even perfect accuracy on the testing dataset. They are plotted below:



A better method than simply fitting and testing is what is known as Leave-p-out cross validation. This is the method that the authors of the paper utilize to improve the machine learning model. This method takes a subset of the data matrix to use as the testing set, and trains the model on the remaining records. This process is repeated p times to guard against over or under-fitting the data. For this problem, it works well since we have 9 subjects, so one subject can be singled out for testing in each of the 9 iterations. Therefore in this instance, every record will be trained 8 times and tested once, ensuring the entire matrix is included in the algorithm.

In the Jupyter notebook here, the ensemble algorithm alone took over 13 hours to run and about 2 hours to fit each individual classifier. The laptop that this was conducted on has only 4GB of RAM, and thus rerunning this using the k-fold method is impractical since the existing method overheated the laptop on multiple occasions. In addition, the authors used F1 score because it is not influenced by the distribution of the classes, which here is clearly not uniform. What we can say is that have room for future improvement both in construction and analysis of the ensemble method. However, our 3 individual classifiers were still constructed in the same fashion, only the split of training and testing that separated them. Therefore, bias reduction is our goal in the future. We could also try alternative methods of NaN reduction, such as elimination, as opposed to our method filling them in with mean column values.

The authors, in their study, used both the ensemble implanted here and a custom one using four different classification techniques. They found their custom model outperformed only the ensemble’s accuracy and F1 score using the decision tree methods, but the individual classifiers as well. Random forest also performed better than the bagging and boosting classifiers. However, they are keen to note that one can more closely examine performance down to individual columns as opposed to the whole dataset. For example, when the random forest model is deployed in free-living conditions, its accuracy declined to just over 50% down from 87%. All of the overall methods, both individual and ensemble, used by the authors showed classification accuracy of over 70% and as high as 85% on the data in question. Given that this is problem for 12 classes, we consider this very good, and especially since the ensemble methods significantly increased both accuracy and F1 score for all methods tried.

References:

Allibhai, E. (2019, August 9). *Building an Ensemble Learning Model Using Scikit-learn*. Medium. <https://towardsdatascience.com/ensemble-learning-using-scikit-learn-85c4531ff86a>

Brownlee, J. (2021, April 26). *How to Develop a Random Forest Ensemble in Python*. Machine Learning Mastery. <https://machinelearningmastery.com/random-forest-ensemble-in-python/>

Calev, A. (2020, May 29). *Time Series Models - PAMAP2 DataSet*. Kaggle. https://www.kaggle.com/avrahamcalev/time-series-models-pamap2-dataset

Chowdhury, A. K., Tjondronegroro, D., Chandran, V., & Trost, S. G. (2017). Ensemble Methods for Classification of Physical Activities from Wrist Accelerometry. *Medicine & Science in Sports & Exercise*, *49*(9), 1965–1973. https://doi.org/10.1249/mss.0000000000001291

Reiss, A. (2012). *UCI Machine Learning Repository: PAMAP2 Physical Activity Monitoring Data Set* [Dataset]. https://archive.ics.uci.edu/ml/datasets/PAMAP2+Physical+Activity+Monitoring