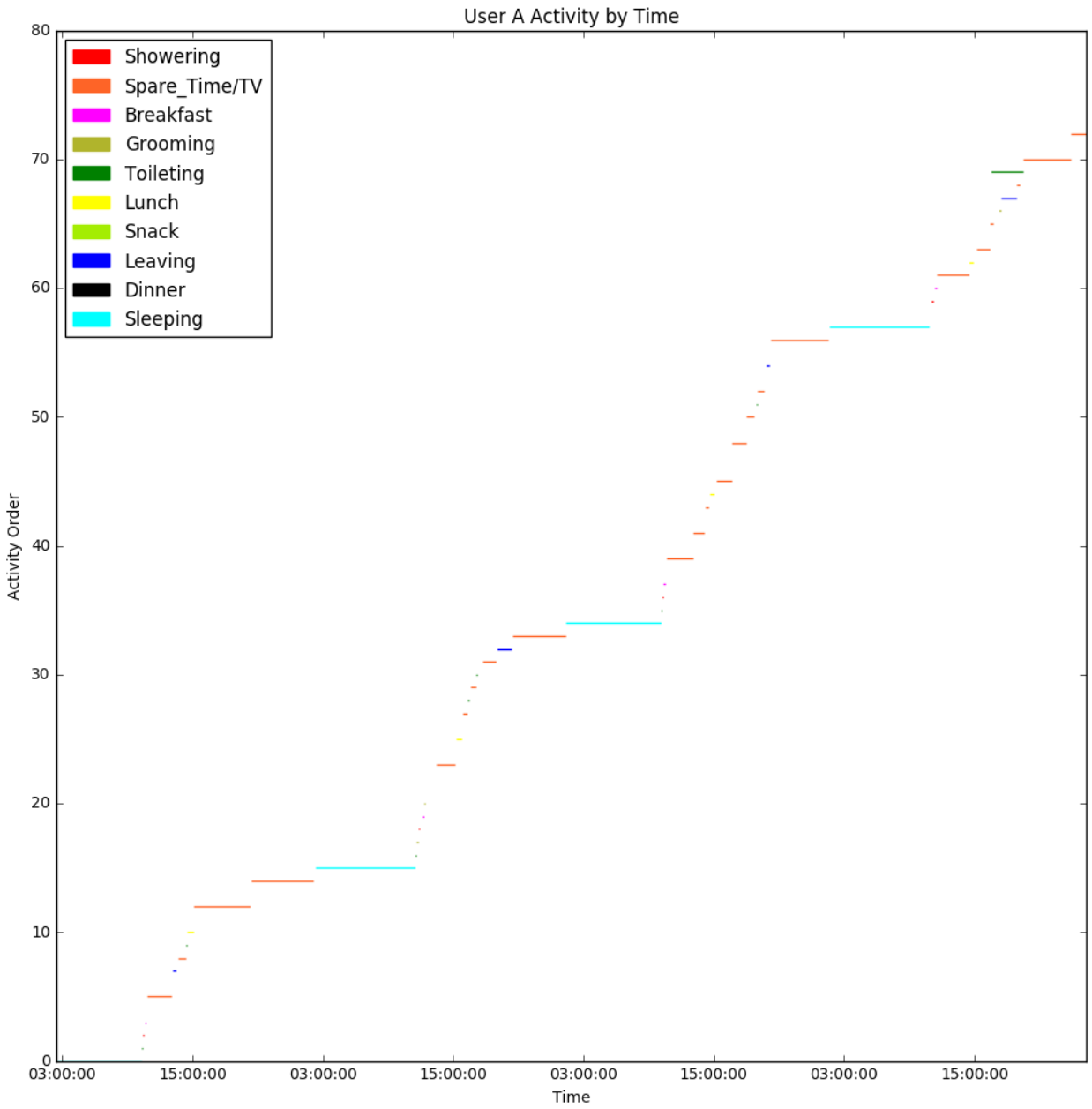
1. Visualize the training data and present it in a easy to understand way.

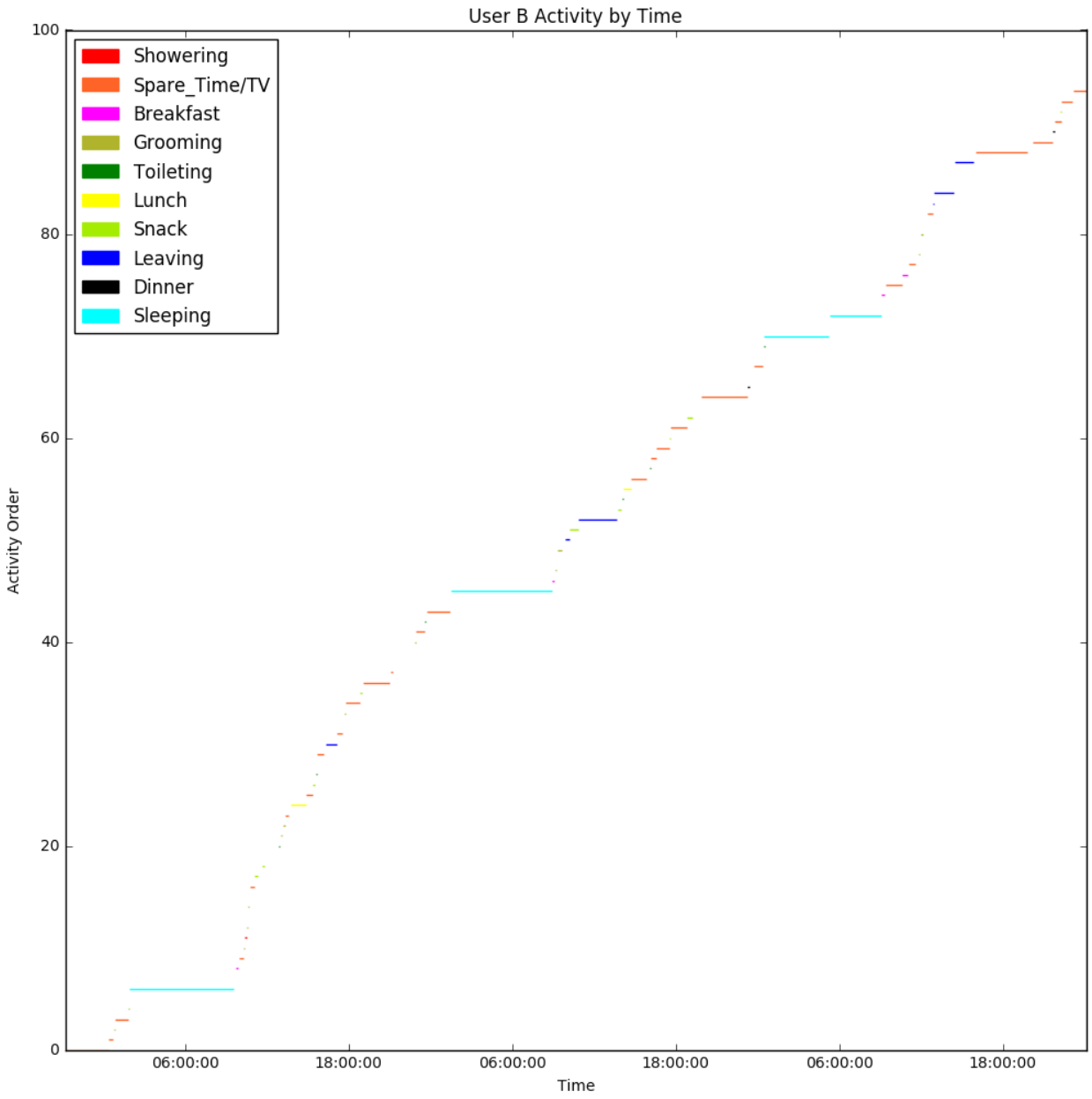
I've chosen to visualize the ADLs of both users as a timeline to represent the natural passage of time, using Gantt charts as the best substitute available. The length of each line represents the length of time the user spends at that activity. The colour represents the activity itself. The y-axis represents time as well as you go from bottom to top.

Here is User A's Gantt chart.



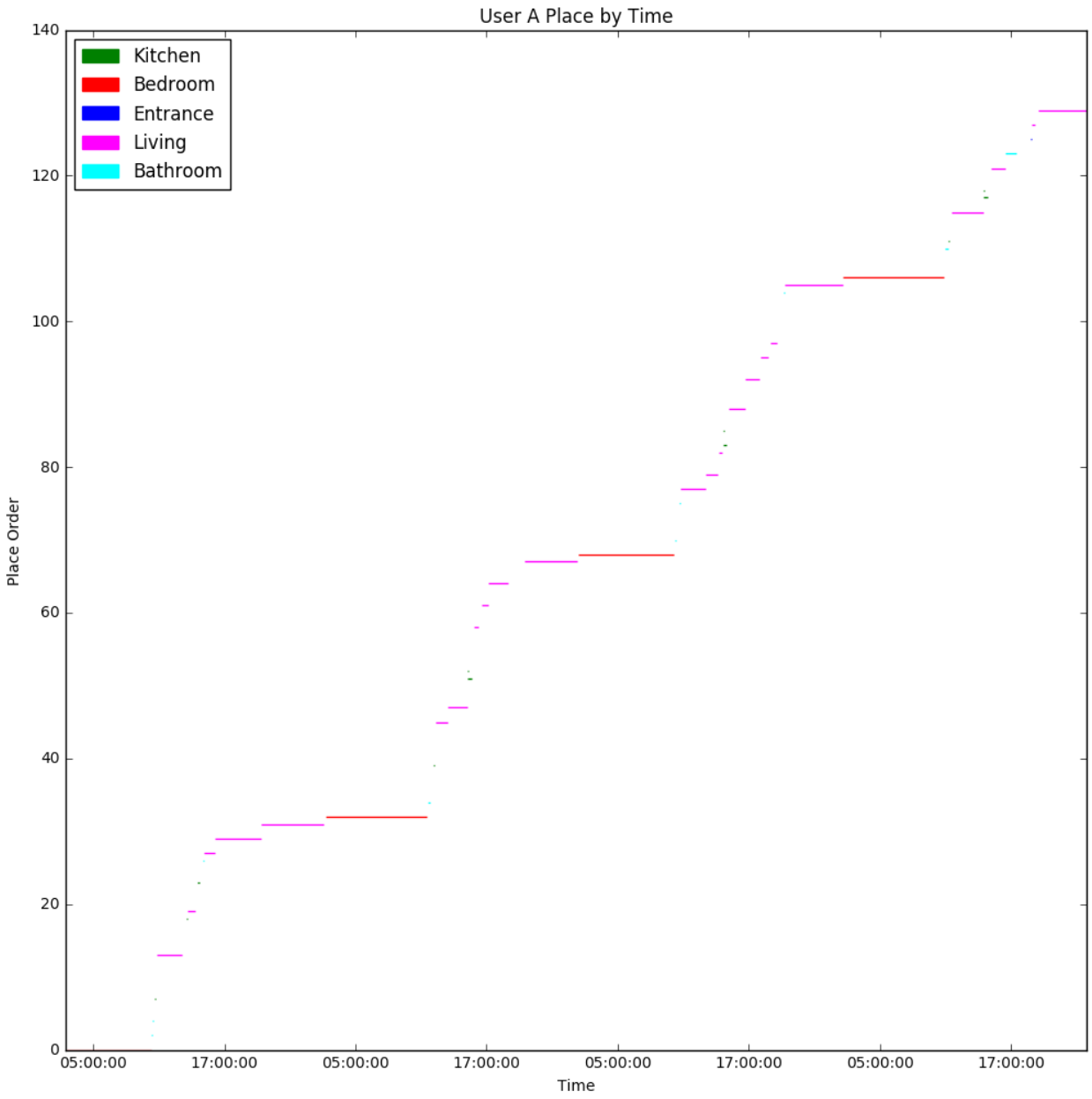
Unsurprisingly User A spends most of her time sleeping and watching TV.

Here's User B's ADL Gantt Chart:



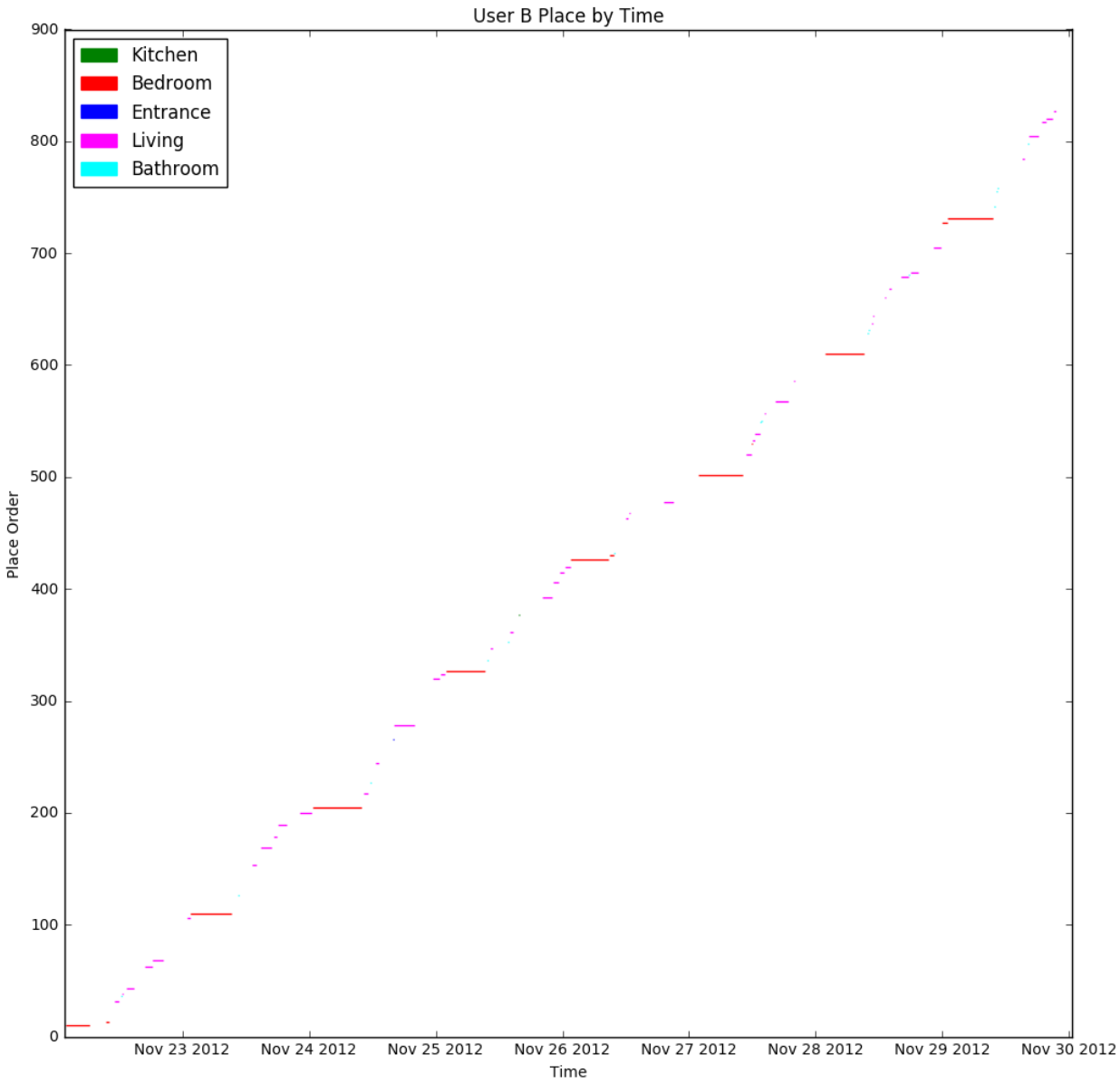
User B spends most of her time sleeping and watching TV as well, but doesn't spend her time watching TV in as long of stretches as User A does.

For sensor data, I've chosen to represent the amount of time a user spends in each place by time. Here's User A's time spent in each place:



This isn't too surprising; User A is spending a lot of time in the bedroom and living room probably because she watches a lot of TV and sleeps a fair amount.

Here's User B's time spent in each place:



User B's time spent in places mirrors her activities as well.

2. Using the training sets, classify the events in 'Classify.txt' and 'ADLs_classifying.txt' between user 'A' and 'B'.

Attached in separate files.

3. Predict the events (ADLs and sensors) that are missing in the datasets 'Classify.txt' and 'ADLs_classifying.txt', for user A.

Attached in separate files. The rows highlighted in Yellow are the predicted missing data points.

4. Explain your choices in algorithms, what are you taking into account, problems you faced and what do you think would be required for a better solution.

General

I found some issues with the data, mainly around fixing data points where End Times occurred before Start Times (which I fixed by flipping them around) and pre-processing the data for ease of use.

Problem 1

I wanted to convey the length of activities associated with each place or activity, but if I had more time I would have plotted each activity/place/location/type by time of day, rather than day of the year, since the data shows some cyclical-ness and the day of the year is largely irrelevant to the activity.

Problem 2

Feature Selection

One of the biggest problems I faced was creating relevant features to add to the model. I settled on the following features:

- Start time (as opposed to start datetime which was given in the initial dataset)
- End time
- Start day of the week (0 for Monday, 6 for Sunday)
- End day of the week

- Duration of activity

For the sensor data I used the following features (encoded in a numerical format):

- Location
- Type
- Place

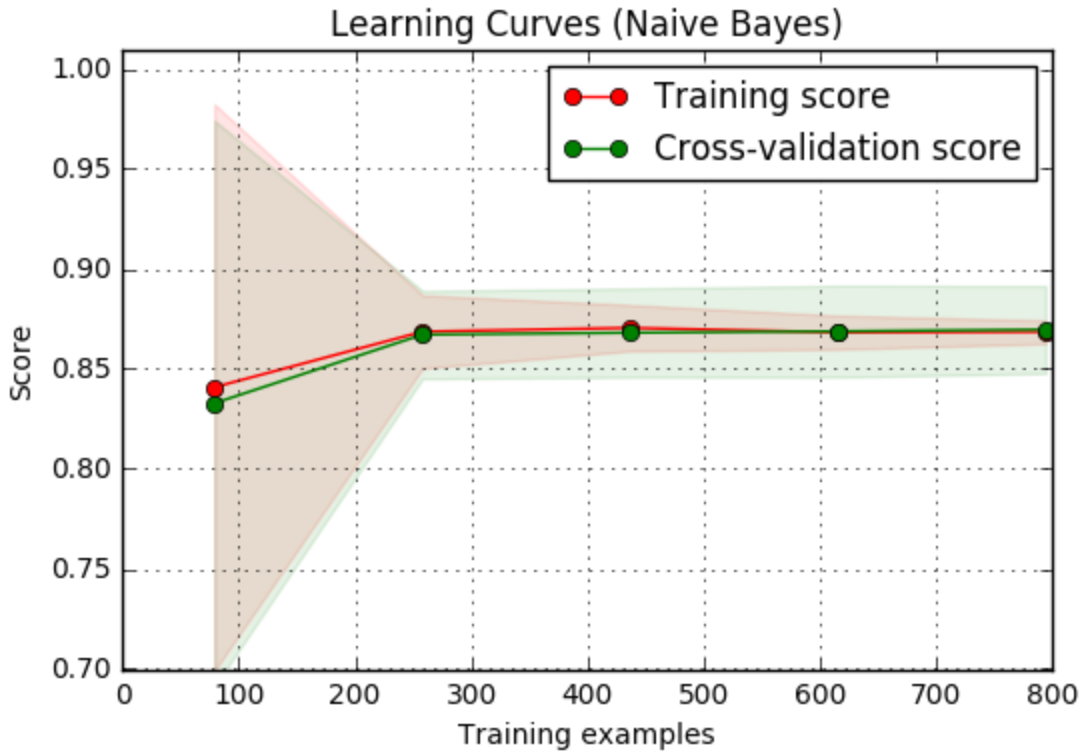
For the ADL data I used the following information (encoded in a numerical format):

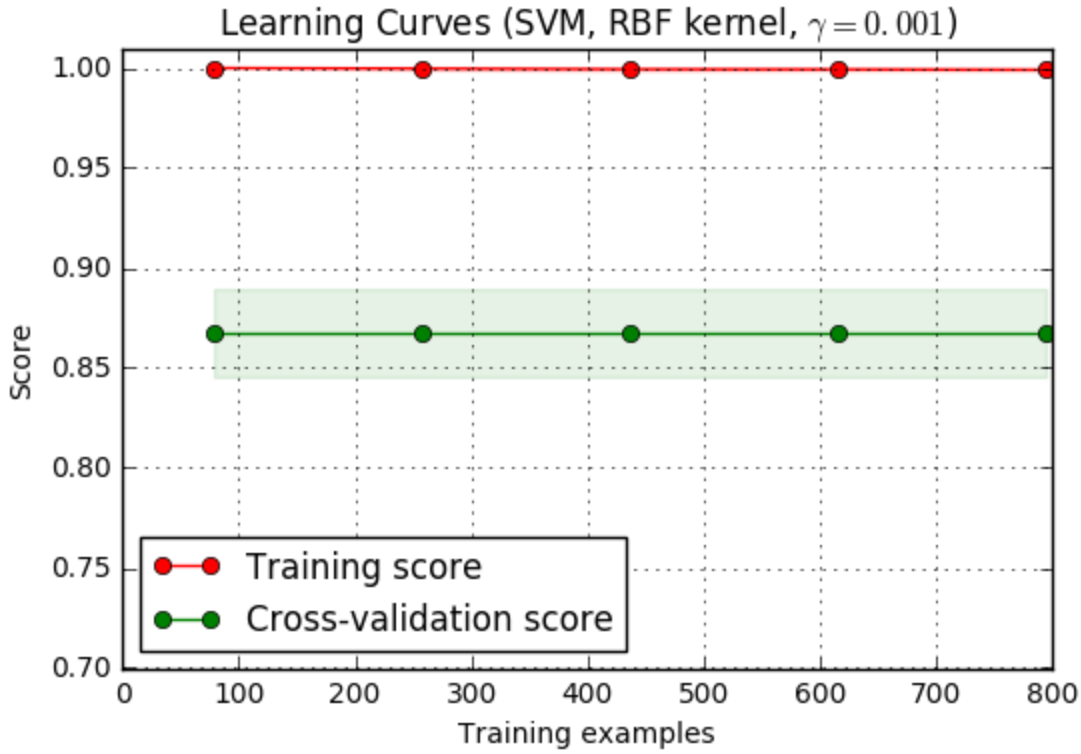
- Activity

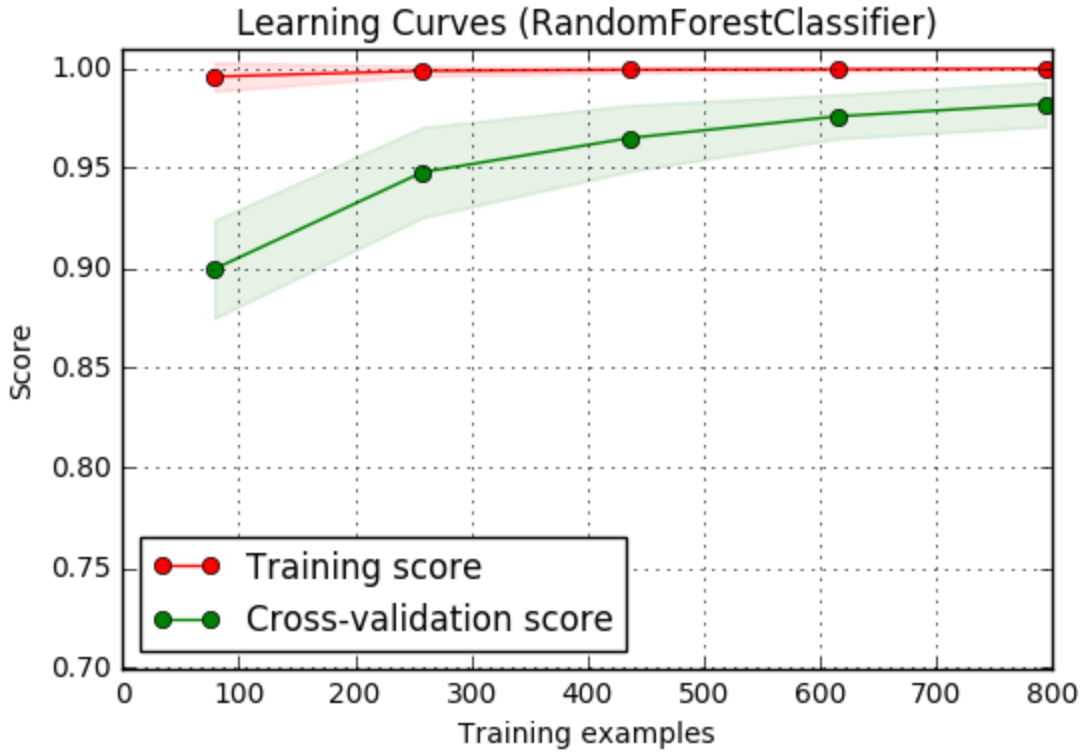
One caveat I made was that each data point is independent of each other. In other words, I assumed that there is no autocorrelation in the data. If I had more time I would have liked to incorporate the timeseries nature of the data better in the algorithm.

Algorithm Selection

Since the task was to predict which ADL/sensor data belonged to which user, the challenge boiled down to evaluating various classifier algorithms. Using 10-fold cross validation I adapted a scikit-learn script to evaluate three algorithms below.




Learning Curves (SVM, RBF kernel, $\gamma = 0.001$)
Score
Training examples
Training score
Cross-validation score
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0
100
200
300
400
500
600
700
800


Learning Curves (RandomForestClassifier)
Score
Training examples
Training score
Cross-validation score
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0
100
200
300
400
500
600
700
800

The clear winner here is the Random Forest Classifier. Not only is its cross-validation performance the best overall, but also the fact that the cross-validation curve converges to the training score is a good sign that the random forest classifier will perform better given more data in the future.

The Random Forest Classifier has a lot of additional benefits as well:

- It's computational less intensive than SVM.
- It provides a ranking of feature importance, aiding in feature engineering.

Problem 3

This was the most challenging tasks I had to deal with.

Get the predicted activities for User A

This was relatively easy seeing as how Problem 2 provided me the predictions. I simply saved those predictions for User A to a file.

Determine which activities are missing

This was a manual process where I determined gaps in activities that were longer than 30 mins (which was determined arbitrarily) and added columns back to the file.

Find a strategy to backfill missing data

This was the most difficult part of all. Normal data imputation methods such as adding the most frequent feature won't work as each activity/place/location/type is associated with the time of day, day of the week, what the user was doing prior, etc.

My best solution was to create a K-nearest neighbours regression using the features selected in Problem 2 and other data points that don't have missing values. The imputed values aren't perfect, but they're a realistic estimate of what the user could be doing at those times.