Q0.

Sheck Wes
Amos Tuwei - atuwei19@amherst.edu - zeuskim
Rob Schwartz - rschwartz19@amherst.edu - rgsthethird
Kaggle Public Leaderboard Score: 25.68561

Q1.

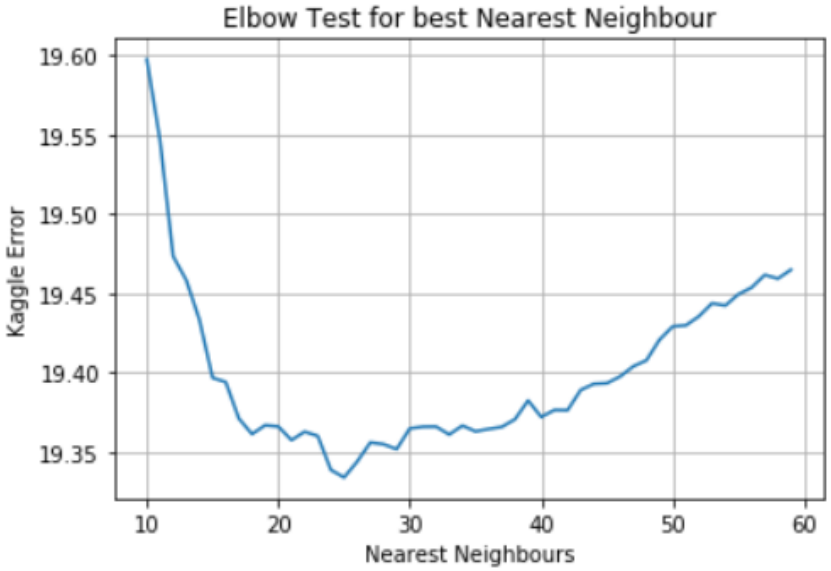
We first read in and format the data from *graph.txt* in a dict entitled "connections". We do the same with the file *posts_train.txt*, creating an array entitled "posts". From "posts", we gather the data that we will use with our learner: the user ID, hour 1, hour 2, hour 3, latitude, longitude, and the number of posts. We then also delete users who have masked their location data ie exist at the null island (Using Wikipedia we confirmed that the only thing existing at this position is a buoy and we are assuming no one lives on it). We also create a dict called "location_dict" with a key of user_id and a value of the [lat, long] location of each user.

With the intention of adding a metric representing a user's friends' locations as features to our training set, we create two new arrays: "med_lats" and "med_longs". For each user in "connections," we get each of that user's friends' locations using "location_dict". We add the median (using the statistics package) of these locations to "med_lats" and "med_longs" if the locations exist (i.e. the user has friends with available locations). Otherwise, we add 0 to "med_lats" and "med_longs" in an effort to maintain the arrays' indices.

Finally, we add the following to "X" as features for each user: hour 1, hour 2, hour 3, number of posts, median friends' latitude, and median friends' longitude. We then scale "X" using a MinMaxScaler.

At this point, we begin to separate "X" and "y" into test and training sets. This is because we need a test set on which we can actually evaluate our errors. We do not know the actual values for the users in the test set.

Regarding how we chose our regressor: we attempted to use multiple different regressors (DecisionTree, kNN, SVM, etc. All of this can be viewed in Regression Exploration) and determined that kNN resulted in the lowest Root Mean Squared Error. Thus, we chose kNN. Regarding the number of neighbors, we ran it for up to 60 neighbors and determined that 25 neighbors was the optimal number to use. See the chart below.



Using sklearn's KNeighborsRegressor with 25 neighbors, we fit the regressor using X_train and y_train**.** We then, in order to predict latitude, trained the learner with inputs of "X_train" and an array of all the latitudes from "y_train," then predict the latitudes for the "y_test" set using "X_test". We do the exact same to predict longitudes for the test set, except we use longitude in every instance that we formerly used latitude. We can combine the determined latitudes and longitudes to determine our final predicted locations for the "y_test" set. We then save it into a .txt file for submission.

Q2.

Well first of all, we would multiply our features and include the averages, median, means of a user's friends of friends. This is in a bid to extract every possible information we can from the data. As a further step in preprocessing, we would check for outliers in the dataset and remove them using methods such as neighbors.LocalOutlierFactor. Then with this, we would run a PCA analysis to determine the predictive power of each feature. This would enable us to use features that result in better prediction. We would also run this in parallel i.e split latitude features from longitude features. This is because some features would be more pertinent to one or the other. For example, time is really helpful in determining the latitude but it would not be as helpful in determining the longitude.
We would then be more intentional on running the cross validation as well as ensembling our models. We would determine under which conditions certain models work best and with that then use those specific models to run on the test set if such conditions ever occurred.
With the above procedure, we believe we will have a really good model ultimately.

Q3.

If we were able to use more data from MyFace+, we would like to incorporate the following features:

1. Interaction with friends' posts

2. Interaction with advertisements
3. Locations of users who also interacted with advertisements
4. Language processing on the posts

We believe that, in terms of interaction with friends' posts, different social circles interact with social media in different ways, and each unique social circle resides in a particular area. Perhaps using a metric that represents interaction with posts as a feature - an action unique to certain people/groups - we can help a learner "strengthen" social circles and therefore pinpoint locations.

Regarding advertisements: even if we don't know a user's location, I expect that MyFace+'s advertisement algorithms will still attempt to provide ads to the user. If a user interacts with a particular advertisement, we can try to identify the user with other users who interact with that advertisement. For instance, if only people from southern Iowa interact with an advertisement for a specific department store and a user without a location also interacts with that ad, we would believe that that user is far more likely to reside in southern Iowa.

We also believe a huge predictor that is not being utilized at the moment is the language in the posts. Depending on which language the post in, this could vastly reduce the range of longitudes/latitudes to consider for a specific user. Consider the Kalenjin language for example, this is rarely spoken outside of the Rift Valley region in Kenya or Swahili for the East African region. Some languages such as English which are global would really add extra information about the locality.