Mackenzie Stein, Brandt Dudziak - Team J3
Amherst College - COSC 247 Machine Learning Final Project

Q0:

1. What is your team name?
   Team J3
2. What is your name, email address, and kaggle username (for both team members)?
   Mackenzie Stein, mstein19@amherst.edu, mstein19
   Brandt Dudziak, bdudziak20@amherst.edu, bdudziak
3. What is your score on the public leaderboard?
   RMSE of 26.62025

Q1: How does your system work? Describe in detail how your predictor makes its predictions. Also describe why you chose what you did.

**Our final system:**

Using the connections file that graphs connections between users and each of their friends on the platform, our system extracts useful information about each user. In particular, we gather the total number of friends each user has and then compute the average of the location (latitude/longitude) of all their friends, saving each of these three values as features in our data set. We expect that if users on our platform have many friends in a certain area, they are likely from that area themselves, and so we thought it would be a worthwhile feature in our predictor. We considered taking the mode of a user's friends' locations (where the locations were discretized, i.e. rounded to the nearest 5, 10, or 25), but in running cross validation, we scored an average of 6-7 points better (based on RMSE scoring) when using averages, which led us to believe that average would be a better feature to use.*

Additionally, the included features that provided each user's first, second, and third most common hour to publish a post required some preprocessing before we could implement them in our model. The first issue is that the values of the feature do not wrap, meaning that although 11PM and 1AM UTC are quite similar times, they would be considered far apart in our model because they are represented as 23 and 1 respectively. To handle this, we group hours into discrete bins based on how time zones fall across different continents. Each bucket has four hours contained in it, and is labeled between 0 and 3. We placed hours 0, 1, 2, and 3 into a bucket labeled 0, hours 4, 5, 6, and 7 into a bucket labeled 1, etc., in order to indicate some similarity across the different hours. The second issue we tried to account for was that these posting hour features included missing values. If a user only posted once, for example, their first most common posting hour would be a real time, 0 to 23, but their second and third most common would both be 25. To account for this, we simply assigned any missing values to be equal to the most common time the user posts. For example, for a user who has posted once, their first, second, and third most common posting hours may be 15, 25, and 25, respectively. We would augment these features in our training or test set to be 15, 15, and 15.

The final piece of data preprocessing that we considered were the users in the training data who were listed as being located at Null Island. We felt that including these in our model would disrupt our model, and so in order to prevent this we simply remove any user located in Null Island from the training set before passing it to our learner.

In summation, our complete list of features is: top 3 most common hours of posting (augmented); number of posts made; number of friends a user has; average latitude of friends; average longitude of friends; most common hour for posting discretized into labeled bins.

After computing all of these features, we passed this augmented data into an MLP neural net regressor. We run cross validation using MSE as a scorer, changing the number of max iterations (we tested 200, 300, and 400) and selecting the value of max iterations based on the lowest score achieved (the scores ranged in the mid-700's). For our training data we found that max_iter=300 frequently produced the lowest score during cross validation. We then take this MLP neural net regressor with the given number of max iterations (300) and predict the latitude and longitude of the users in our test set. While we could have adjusted many of the parameters, we decided to stop while we were ahead because we did not want to run the risk of overfitting. We are confident that our model is intuitive, uses our data well, and is not overly complicated, and therefore effectively captures the relationship between a user's location and their features represented in the model.

*We thought about using average vs using mode a lot when it came to how to use information of users' friends locations. We ultimately decided that average would be better, given our cross validation scores as well as the following intuition: Say someone has 1 friend in LA and 1 friend in Paris. If we take the average of the latitude and longitude of these locations, we'll get somewhere in the middle; if we take the mode, we'll randomly select one location or the other (or, we'd take the latitude of one and longitude of the other). Then, if we predict that this user is at the average locations, we're probably wrong: it's much more likely that they're in LA or Paris, not the middle of the Atlantic. But, if we predict the user is at the mode, then we're either (almost) exactly correct, or very very wrong (and about twice as wrong than predicting they're in the Atlantic). So, essentially, while we know that using the mode should be better in theory, we feel that when it comes to the error measure that we're considering, it made more sense to use the average instead. In particular, if we had more time and perhaps more information, we could "intelligently" select the mode: figure out the best way to round the latitude and longitude of friends' locations so that the mode is a more accurate value, not just a randomly selected one out of the bunch; or use other information, perhaps compare the hours of posting, to break ties about which mode to choose; etc.

**Steps we took toward our final product:**

Our initial intuition was to precompute various kernel matrices that our models could use to make their predictions more robust. Among the kernels we wanted to build was one that measures two users' similarity based on how close they are connected on the connections graph. We computed users' distances using Dijkstra's algorithm, so that i.e. if 2 users were friends they had a distance of 1, but if they were not friends but had a friend in common then they had a distance of 2. We would then take the inverse of these distances and use them as a measure of similarity between users. We also thought about different ways to weight these distances, primarily based on how many total friends the users have. That is, if one user has only 1 friend, they are likely closely related to that user, compared to if a user has 2,000 friends then the average "similarity" to each of them is much lower, since they are likely closer to some and farther from others.

Other kernels we included compared the number of friends 2 users had and the number of posts 2 users made. We ended up using these as features in a slightly simpler way than our designed kernel. We also thought of some more complex ways to relate posting hours. That is, we compared all 3 of one user's top hours to all 3 of another user's top hours, pairwise (so, 9 pairs of hours) and weighted them by which hour number they were for each user (i.e. weight the (hour1, hour1) pair greater than the (hour3, hour3) pair, and put the (hour1, hour3) and (hour3, hour1) pairs somewhere in between). This kernel also handled the case where a user has 25 as a top posting hour, given that this an indication of a missing piece of information.

Unfortunately we were unable to use any of these kernels because they took too much time, processing power, and memory to compute. We attempted to find other ways to incorporate this information in our model, using augmented and preprocessed features instead.

We tried a number of different regression models and classifiers with different parameters using grid search, including kNN with k = 1, 3, 5, 7, 9, 11, an RBF SVM with gammas 1, 10, 50, 100, and Decision Trees with max depths of 3, 5, 8, 10, 11, 12 (depending on the number of features being used). We then used cross validation to evaluate each model. MLP ended up giving us the best model, when each was scored using MSE. Originally we used some additional models to predict features we believed could have been useful, like predicting latitude first and then including the predictions as a feature when predicting longitude. We also had a model that classified points either as east or west of 25° W (longitude of -25). We believed, because large collections of points in the training data were placed in North America or Europe, that classifying testing points as in either region would lead to a more accurate predicting for longitude. We found this wasn't true, as points that were incorrectly classified would then be placed in the wrong part of the world and would face substantial loss. A final tactic that we tried was to produce a classifier which used all training data to predict whether a test point would be at null island or not; we would then use these predictions to either note the location as (0,0) or learn a model (with only not-Null-Island training data) to predict the actual locations of each of the test points.

Q2: If you had this same data, but an extra six months you could dedicate to building the best prediction system you could, what would you do?

First and foremost, we would have liked to try to complete the things we attempted but failed at. For example, as mentioned in Q1, we tried to compute a few kernel matrices based on kernel functions we designed, but ultimately realized that

computing them took a prohibitive amount of time, computation resources, and/or memory space. If we had an extra many hours available to us, we would love to be able to finish computing those kernels and see if they helped us compute the similarity of users. Of course the first step of this would be to see if we could streamline our computations at all and make the whole thing take less time in the first place.

Beyond that, we would focus on how to get as much information as possible from the features we have. We did extract information about each user's friends, their friends' locations, and the most common hour they post at, but could certainly do more here. One idea is to use the mode of friends' locations, rather than the average. As mentioned above, we feel that this would likely be a better piece of information to use, as long as we can intelligently handle edge cases, such as that where a user's friends are all from very different locations - so, the first step would be to find better ways to choose the mode in these cases, which would likely come from using the users' posting hours as a tiebreaker (that is, more similar hours likely means they're in a similar location) or finding a better way to discretize the locations. In particular, we tried rounding to the nearest 5, 10, or 25, but presumably there would be a "best" choice for this, which we could perhaps find from a sort of cross validation. We could also try taking the top 3 most common friends' locations, or taking some sort of weighted sum of (mode + average) locations, or use both mode and average as different features.

Another idea is to find a more sophisticated way to encode the idea "hour 0 is close to hour 23", and find a way to relate top hours posted to friends' locations - i.e., if all my friends live in the same place and post at the same time, but I post at times 3 hours ahead of them, can we parse that somehow and say I am at "longitude of 3 hours ahead"? Finally, we'd like to work in some information about "friends of friends:" essentially try to parse the same information about our friends and our friends' friends, and perhaps go a few layers into this (i.e. examine everyone that a user can connect to in 4 steps or less). This last one could cause us problems, especially depending on how far we reach, in that it may cause us to include information that is not actually relevant to predicting a user's location if we use people who are too far away from a given user.

Another thing we would have liked to improve upon is our consideration of null island. Right now we simply remove the users from the training set whose given (lat,lon) = (0,0), and use the remaining data to actually learn with. However, we feel like there must be a better way to incorporate this. One idea we had, as mentioned, which we did not have time to fully test, was to first classify the testing data as either being at null island or not, and then prediction "real" locations for only users who we believe are not at null island. This would require some thought about how to create a classifier for "yes at null island vs. not at null island," which could ultimately prove more difficult than beneficial. Another idea we had is to first try to predict the actual locations of users in the training data that show up as being at null island, and then use this version of the training set to learn a model. (Of course, if we were doing this in ~the real world~ we would also want to think about whether we want to predict people at null island or not: in some ways this is useful information, since we can then assume certain things about our users who spoof their location to appear at null island; but in other ways it's far less useful, as it does not allow us to accurately track a user's location and i.e. present them with targeted ads.)

We also never gave much attention to the idea of outliers. There are likely some users in the test set that are outliers, or certain clusters of users, or other various patterns of the underlying dataset. We never put much thought into how to detect or handle these things, as that would have likely taken a good amount of effort to both detect and then handle, but if we had more time that could be a very beneficial bit of information to parse.

Finally, we would like to test more types of learners and more parameters for our learners through cross validation. We ultimately ended up simply not having much time to do this, but recognize that this is the best way to tune a prediction model and figure out what the best type of learner would be as well as which parameters work best for the given learner. We certainly noticed along the way as we were trying different things that running cross validation with different parameters for our learners was very important to choosing the best model. Cross validation is also one way for us to prevent overfitting, which is a constant concern when working with machine learning algorithms.

Q3: If you could get more data from MyFace+ to help infer user location, what would you want and how would you use it? Focus on data that a company like MyFace+ probably already collects anyway (e.g., login times, post content, number of seconds the user has each individual post on screen (yes, everybody collects this), links clicked, etc.).

*Note: for this section we're assuming MyFace+ is similar to Facebook, and thus has features such as groups that a user can be a part of, events that the user can "attend," the ability to publish personal information such as schools attended or job experience, and the ability to directly message other users.*

As mentioned above, we're interested in looking at events that a user has been invited to or attended, and schools and jobs listed on their profile. Often events, schools, and jobs have locations attached to them, which we could parse and use as new features to help predict their current location. We would also want to think about how to weight each of these elements, since older ones are less relevant. That is, my high school might be less relevant than my college which might be less relevant than the job I currently have if I have graduated college, or an event I attended yesterday might be more important than an event I attended 7 years ago, etc.

Another feature that we could use is groups that a user is part of. This would potentially give us a location, if the group is attached to a location (or company, school, etc. which would have a location). It would also give us another set of users which the given user might be connected to in some way, and we could proceed to use these users' information in our predictions.

One other important feature would be the number and times of messages sent to any given user. This could tell us not only who someone is friends with, but give us a sense of "how close" they are to each other: if they message each other a lot, they're likely close friends, but if they have never messaged each other, then they aren't as close. We could use this metric as a way to weight a user's friends' locations, hours, etc. - any feature we extract from them. In a broader sense, anything that would give us a sense of two users' closeness could be useful for this goal: how many photos or posts they've both been tagged in, how many events they've both attended or groups they're both part of, etc.

Moving on to slightly more obscure pieces of data:

If MyFace+ collects information about IP addresses, this could be quite useful, assuming we could also find the location of the IP addresses, which we could then use as something that would likely closely correspond to the user's location. (This sort of feels like cheating, but since I don't entirely understand how IP addresses work and how a company like MyFace+ collects user's location information, I'll mention it anyway. Also, I know people can spoof their IP address, so this would be a consideration when using this information, but it could still be useful.)

We could also try to parse the language that a user has set as their default language on their account or on their device. This would not have perfect correlation given the spread of languages throughout the world, but it could steer us towards locating them in certain regions in the world with higher probability or it could act as a tiebreaker, according to where the language is most commonly spoken.

We also feel that collecting information about which ads a user watches or the specific content of the ads watched could be useful. Often ads are targeted to be presented based on location, which could help us narrow down a region in which we believe the user will be. For example, ads seen in the US would likely be quite different than ads seen in Hungary. Or, further, we know that during election season different regions of the country are presented with very different and sometimes very localized ads, which could be quite useful information.

Another interesting feature to consider would be other websites that a user visits, or what links a user clicks on. This again could give us an indication of different regions that they are located in, depending on how much information we can parse from these other websites.