Predicting Social Network Check-in Locations: Noise Impact Reduction for Classification

Elnaz Jedari Fathi
Department of Computer Science
Southern Illinois University
Carbondale, Illinois 62901
e.fathi@siu.edu

Shahram Rahimi
Department of Computer Science
Southern Illinois University
Carbondale, Illinois 62901
rahimi@cs.siu.edu

Dunren Che
Department of Computer Science
Southern Illinois University
Carbondale, Illinois 62901
dche@cs.siu.edu

***Abstract*—Since 2010, Facebook has entered the self-reported positioning world by providing the check-in service. This service allows users to share their physical location. Over the years, big datasets of recorded check-ins have been collected with increasing popularity of social networks. Analyzing the check-in datasets reveals valuable information and patterns in users' check-in behavior as well as places' check-in history. In this work, we intend to create a prediction model to predict the next check-in place only based on places history and with no reference to individual users. To this end, we learned that the check-in data has a high level of noise in location coordinates. The main research objective of this work is how to leverage a noise impact reduction technique to enhance performance of prediction model. We designed and developed our own noise handling mechanism to deal with feature noise. This work represents how the performance of predictors is enhanced by minimizing noise impacts.**

I. Introduction

Many social networking services, such as Foursquare, Twitter and Facebook, provide users an option referred to as self-reported positioning [1], [2] which is known as ”check-in” among users. Facebook check-in data is one of the largest collection of check-in activities. Analytical results of mining these location-based datasets reveal patterns of user behavior and place check-in history. Extracting this knowledge and predicting location check-in traits are utilized in business and financial decisions [3], but also are helpful in many scientific pursuits as well.

Check-in prediction models can be generated by analyzing either users' behavior [4] or places' history. Both approaches generate a prediction model for check-in locations. In this paper, we study a big dataset of recorded check-ins in Facebook application based on place history and with no concept of person [5]. There are about 29 million rows in the dataset and each row represents a check-in event. We intend to classify all check-in instances into categories which are places. Therefore, this is a multi-classification problem with a large number of predefined classes (categories). There are various machine learning algorithms to solve classification problems. However, most algorithms require well-defined data to efficiently learn and extract patterns from training dataset. Real-world datasets are influenced by many factors that can affect the quality of the dataset [6], [7], [8]. The source of data is one of the significant factors [9]. In check-in datasets, the source of data is location signals, coming from users' mobile devices where not 100% are accurate. One of the reasons is that users desire to check in from different distances away from the place's physical location. Additionally, GPS signals received by mobile devices are not always consistent. These factors along with other similar factors prevent the data from always being well-defined, thus noise and outlier instances appear in datasets. The presence of noise in data mining datasets mislead the learning process of machine learning algorithms and accordingly reduces the performance of the generated prediction model. Hence, variable noise handling approaches are studied in the literature.

In this work, we present a noise detection and cleansing approach for location-based data classifications. We show how the predictors' performance is enhanced by applying our noise impact reduction mechanism. After enhancing the dataset quality, a machine learning algorithm can be trained on the data and generate a predictive model for the target value. Despite the selected learning algorithm, the generated model should provide a better performance compared to models trained in noisy environment.

This work is organized into the following sections. Section II presents a brief overview on noise impacts and handling approaches in classification tasks along with related works. In section III the dataset is described and its features and characteristics are explored. Section IV provides the preprocessing noise handling technique and describes how we clean noise instances prior to the training process. In section V, Random Forest classification methodology is examined to provide performance evaluation of the noise cleansing technique. Section VI concludes the study.

II. Noise Impacts and Handling Approaches in Literature

Check-in datasets include location-based data that are typically polluted with high levels of noise because their data collection source comes from users mobile devices. Inconsistent GPS signals and users desire to check in from different distances away from the places are two factors causing location noise appearance in dataset. As a result, it becomes more difficult to define the borderline of the places (classes) while classification process tightly depends on how well the borderline of classes are defined. Classification algorithms can simply fail in the presence of large amount of noise [8].

Experiments in the literature show the presence of noise in data mining datasets misleads the learning process of machine learning algorithms and accordingly reduces the performance of the generated predictive model. The consequence of noise on classification performances is the most frequently reported issue [8]. Hence, noise handling approaches in classification are widely studied in the literature. Some algorithm such as Random Forest [10] or Decision Tree [11] with pruning can justify their learning process in the presence of noise; however, in real world with big data that are normally complicated by inaccurate and noisy instances, they barely can overcome the noise misleading impacts. Therefore, a preprocessing technique or outlier detection function normally is required to clean noise instances before or during the training process.

In [8], authors provide a comprehensive investigation on noise handling methodologies in literature. They distinguish three types of methods to deal with noise: 1) Noise-robust algorithms, 2) Noise-tolerant algorithms, and 3) Noise cleansing methods. The first two approaches deal with noise in learning algorithm during training process and normally results in higher time and complexity cost. On the other hand, the third type approaches are preprocessing filtering methods which detect noise instances and deal with them before sending the data to training process. In this work, the third approach is utilized and a noise cleansing method is introduced which is a fast and scalable noise preprocessing method. This method is applicable to large datasets such as the one used in this research.

An example of preprocessing noise cleansing method is classification filtering approach which is utilized in [12]. Two different classifiers are trained and combined to vote for misleading instances. Misclassified instances in both classifiers are marked as noise and eliminated from training set. The noise elimination cause loss of the information stored in other features. In high noise level datasets, the information loss becomes more critical. Therefore, our goal is to not eliminate detected noise instances; instead, we replace noise's misleading values with correct or representative values.

Moreover, there are many kNN-based noise detection algorithms in literature that are based on nearest neighbor editing rules. A survey of these algorithms is included in [13]. Summarized kNN-based algorithms in this work are basically a modified version of ENN [14] algorithm. As it is addressed in [13], the major problem of these algorithms is their usage of memory and storage in the process of noise detection and also the need to reduce the number of flagged instances by providing new additional rules.

A quantitative study of noise is accomplished in [9] that is been attracted by many researchers. The authors of [9] consider preprocessing mechanisms a more reasonable solution to handle noise instances. They focus on feature noise rather than class noise in their study. In this investigation, several important conclusions are drawn to deal with noise in datasets. One of the most important conclusions is the influence of feature noise on system performance is different depending on which attribute is noisy. If the noisy feature is highly correlated with the target feature, then the negative impacts of feature noise will be higher. Therefore, the authors recommend to design our own noise handling approach to improve the data quality. We follow their advice and develop a feature noise handling from our own perspective.

III. Dataset Characterization

In this section we describe the check-in dataset and explore its features and characteristics. The dataset is provided by Facebook, in one of the Kaggle competitions [5]. Facebook created a 10 kilometer by 10 kilometer artificial world consisting of over 108,000 places distinguished by an identification number which is stored in *place_id* future. There is no concept of person in the dataset and each row of dataset represents a check-in instance recorded within the border of provided area. For each instance, other information is stored along with the identification of place, such as check-in time, and location information of the device used to check in. An overview of the dataset attributes along with some samples are shown in Table I.

TABLE I
Overview of features in check-in dataset

row_id	x	y	accuracy	time	place_id
0	0.7941	9.0809	54	470702	8523065625
1	5.9567	4.7968	13	186555	1757726713
2	8.3078	7.7968	74	322648	1137537235
3	7.3665	2.5165	65	704587	6567393236

The check-in dataset is fabricated by Facebook to resemble location signals coming from mobile devices [5], which is complicated with noise and outlier instances. The location information is expressed in physical coordinates of x and y tuples simulating latitude and longitude of the location. Consider a public business place in a city. Apparently, all users do not check in from the same distance away from this particular place. Thus check-in instances represent different coordinates for each particular place and we do not have access to the real coordinates of places.

There is an eccentric characteristic of location coordinates in the dataset which is a considerable difference between x spanning range and y spanning range for individual places. If we magnify individual places' coordinates, as it is indicated in Figure 1, we will notice that the range of x is mainly expanded because of outliers. We do not know why the outliers have happened mainly to the x feature and rarely to the y feature. Ultimately, this difference does not influence

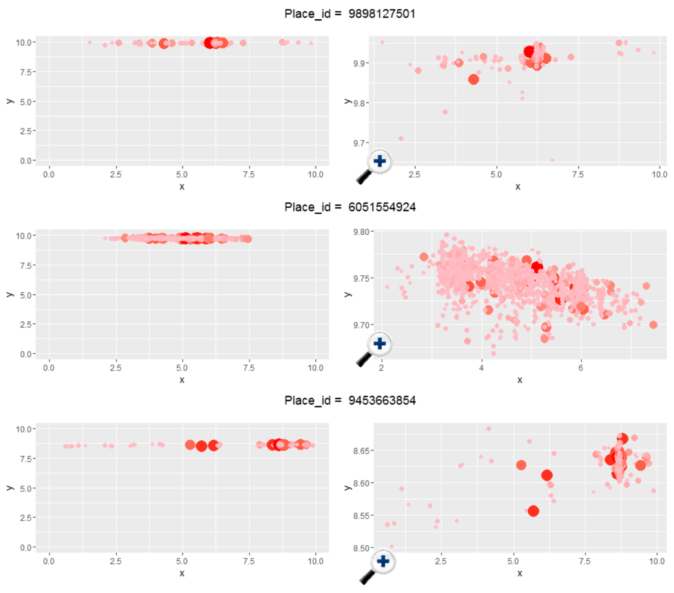

Fig. 1. Check-in coordinates for random places represented in separate planes. A magnified representation is added, as well. The larger points with darker color represent a higher value of accuracy.

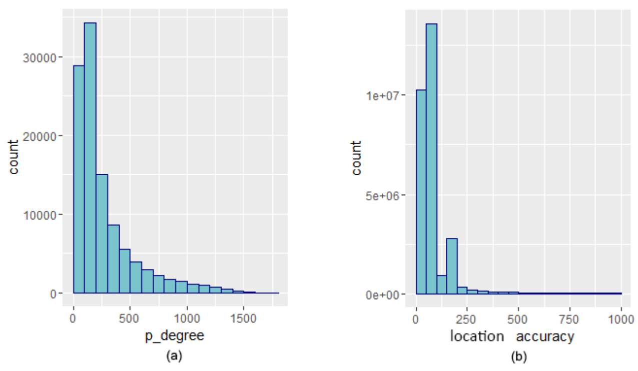

Fig. 2. Distribution diagrams of popularity degree and location accuracy. Diagram (a) shows the number of places in the dataset based on popularity degree, and diagram (b) indicates the distribution of location accuracy values among all check-in instances.

our classification approach, but it benefits the algorithm. Less correlated features better serve the classification algorithms training.

The feature accuracy contains information about location accuracy for each recorded check-in; however, this feature is intentionally left vague [5]. A hypothesis about the accuracy representation is the veracity degree of calculated physical location of users in the GPS system. Factors like weather condition, or surrounding buildings can disrupt the communication between satellites and GPS signal receivers in mobile devices. Therefore, the accuracy of estimated locations of user may vary in the presence of named factors. In Figure 1, larger points with darker color represent higher accuracy. By considering our hypothesis, how can we explain coordinates with very high accuracy that are located very far from the majority of other coordinates? One possible reasoning is a human-side error has happened. The user has checked in from a longer distance away from the physical location despite accurate computation of user location in the GPS system. Therefore, outliers appear on the plane and we cannot count on coordinates with high accuracy to locate actual location of *place_id*s. According to the histogram of accuracy distribution in Figure 2 (b) , the majority of check-in instances represent a value of accuracy below 200 while the range of values for accuracy is 1 to 1,033. Only 5.3% of instances are recorded with an accuracy higher than 200.

Lets look at an interesting feature of the dataset. Places where a large number of users check in are popular places with the most number of repetition in the dataset; therefore, a higher number of instances represent them in the dataset. To demonstrate the distribution of check-in frequencies for individual *place_id*s, we define *p_degree* variable as popularity degree. This variable spans from 1 to 1,849 which simply means the place with only one check-in record has a popularity degree of 1, and the most popular place with the largest number of check-ins has a popularity degree of 1,849. Figure 2 (a) illustrates the distribution of places' popularity degree on the whole dataset. The diagram indicates that the majority of places have been checked in less than 500 times while there are places that are checked in as high as 1,800 times.

The other important feature in our check-in dataset is the *time* feature which encompasses integer values spanning from 1 to 786,239. Examining some crucial *time* values, particularly the largest recorded one, we are convinced that the integer number represents the time interval and it is calculated in minutes. The largest value of *time* in the dataset is 786,239 which represents exactly one year and a half minus one minute. We converted the integer numbers stored in *time* feature into timestamp, then break it down into new features such as hour, day, and month.

IV. Data Preprocessing and Noise Handling

This section represents our methodology for dealing with feature noise in location-based datasets. We select a random 400 meters by 200 meters area and represent the locations of all check-in events in Figure 3 to demonstrate the level of noise. In the figure, the locations of check-in events are represented with different colors for different places. Obviously, the noise level of this dataset is significantly high and place borders are very fuzzy. To reduce feature noise impacts on classification learning algorithm, we implemented a practical method to correct feature noise instances and avoided eliminating them from the dataset. We believe that eliminating noise instances, especially when the level of noise is high, not only does not improve the algorithm performance, but also it may mislead the learning process. Noise instances contain important information in their other features, whereas, eliminating them causes a noticeable information loss. As a result, features importance level changes in decision-

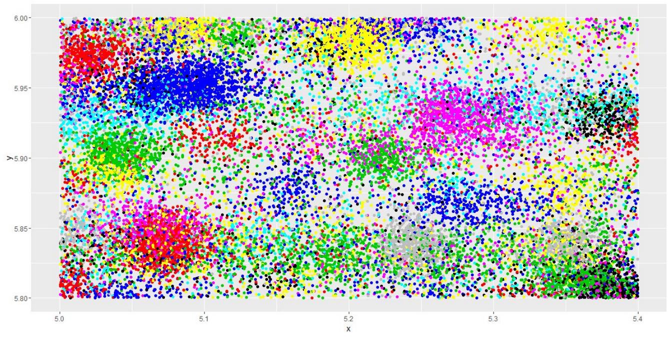
Fig. 3. All recorded check-ins in a random area of 400 meters by 200 meters. Different colors represent different places

tree-based algorithms such as Random Forest [10]. Feature importance level has a key role in splitting nodes decisions. Hence, instead of eliminating noise, we cleaned them by replacing feature noise inaccurate values with representative values and reduced their misleading impacts on the learning algorithm. The following sections demonstrates our feature noise cleaning approach.

A. Cleansing Mislabeled Instances

We clean feature noise instances in the check-in dataset to enhance the quality of training dataset. The goal of this process is to reduce the impacts of feature noise on our selected learning algorithm, Random Forest [10]. We believe that this step has a significant influence on the performance of generated model.
Noise appearance in the dataset is a consequence of mislabeled instances. There are two general reasons why mislabeling occurs. The first reason is the class feature has been incorrectly labeled. The second reason is the value of one or more other features are erroneous resulting in the instance being placed into a wrong category. Depending on the reason of mislabeling occurrence, one of two cleansing approaches may be employed. Figure 4 illustrates the two approaches. For the first reason of mislabeling, the noise instances need to be relabeled so they will be categorized in the right class. For instance, in Figure 4 (a) mislabeled points are corrected based on an estimated classification border. In this approach, an approximate classification border is required which can be estimated with a density-based classification method as an example. On the other hand, if the second reason for mislabeling occurs, the second approach, illustrated in Figure 4 (b), can be applied to cleanse mislabeled records. This approach cleanses noise instances by correcting erroneous x and y values which changes the physical location of instances without relabeling them. Similar to the first approach, a hypothetical barrier is required to evaluate the correctness of x and y. In check-in dataset, the error in x and y features mainly appears because of users' desire to check-in from a large distance away from the place's physical location, or the GPS system miscalculation of user actual location coordinates. Therefore, changing the value

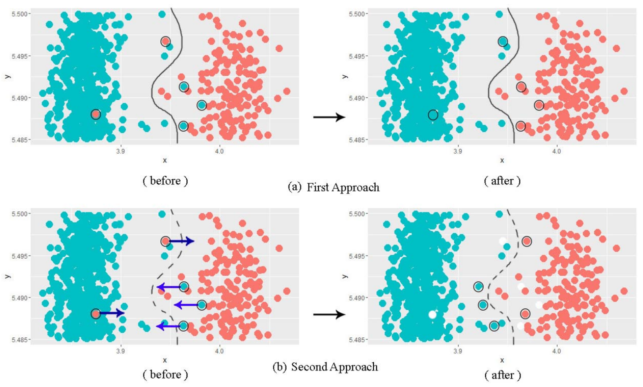

Fig. 4. Noise cleansing approaches. (a) shows relabeling class errors and (b) illustrates correcting the values of features other than class feature

of x and y does not ruin the check-in instances.

B. Statistical Noise Cleansing Method

In this section, we represent how the approximate guideline for each place is determined individually so the outlier records of each place can be separated from correct instances. This module was executed using 8 CPUs and 64G memory utilizing R vectorization property. The vectorization property speeds up arithmetic calculations by avoiding the use of programming loops. When the outliers (or noise instances) of places were recognized, we cleaned their erroneous x and y values by replacing them with representative values. Statistics and probability distribution tools were employed to determine the places' guidelines. R vectorization property was utilized to correct the erroneous values with representative values.
Statistics quartile concept was utilized to find places' guidelines. The first and third quartile of both x and y values were individually calculated for each place. Every point smaller than first quartile or larger than third quartile in their x and y value was marked as outlier. In fact, all the values between the 25th percentile and 75th percentile of both x and y were considered as correct values and everything else was erroneous. Afterward, the means of x and y correct values were calculated separately to be replaced as the representative value for erroneous values. Figure 5 indicates all check-in instances for three random places while their outliers are calculated with this technique and are displayed with different color. This noise (outlier) detection technique recognizes over 7.5 million noise instances in the whole dataset which is 26.28% percent of the data.
We individually calculated first quartile, third quartile, and mean of x and y for each place utilizing R vector property which took 14 minutes to compute for the whole dataset using the platformdescribed earlier. Then we injected theses values to the original dataset as new features. This step requires a merge function which took 20 minutes to run. Afterwards, we flagged outliers and stored them in a new feature called $isOutlier$ and applied conditional statements which is implemented by R vectorization property. Consider

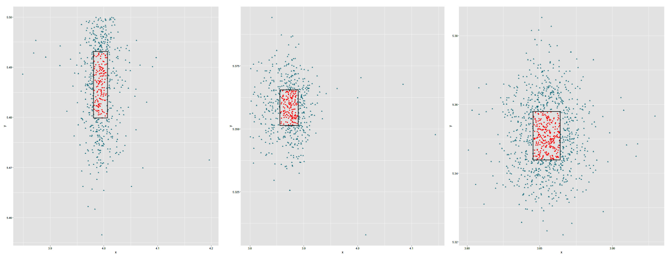

Fig. 5. Outliers of three random places displayed with different color. These outliers are marked based on the first and third quartiles guidelines.

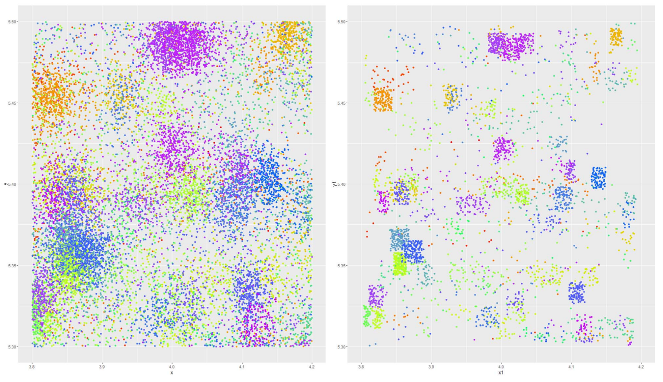

Fig. 6. Noise reduction preprocessing effect on a sample area with multiple places. Places are represented with different color.

Q_1x and Q_3x as first and third quartiles of *x*, and Q_1y and Q_3y as first and third quartiles of *y*. The conditional statements to set the outlier flag would be:

$$\begin{aligned} isOutlier = ![\ (x > Q_1x)\ \&\ (x < Q_3x) \\ \&\ (y > Q_1y)\ \&\ (y < Q_3y)\ ] \end{aligned} \tag{1}$$

True value of *isOutlier* indicates if the check-in instance is an outlier for the represented place. The accumulated time for labeling noise instances was 36 minutes using 8 CPU cores. The accumulated time was 39 minutes to accomplish noise detection and cleansing processes on the whole dataset, which contains over 29 million rows. The simplicity of the approach and vectorization property of R programming language are the major runtime efficiency factors in the presented approach. This method is applicable on other location-based noises.

V. Implementation and Evaluation

In this section, the effectiveness of our preprocessing feature noise cleansing technique is evaluated. We generated two Random Forest models, one was trained with noisy location coordinates, and the other was trained with cleansed location coordinates. We evaluated the effectiveness of our presented noise cleansing approach by comparing the performance of the two generated models. The models' evaluation is based on predicting the target feature in the test dataset and measuring the difference between predicted values and the expected values, which represents the Mean Squares Error (MSE).

Random Forest models were trained utilizing R *randomForest* package which were executed using 16 CPU cores of Bigdog cluster. The models were trained on three distinct samples drawn from the original training dataset. Stratified sampling technique was used to draw samples from places with variable popularity degree, then regular samples were drawn from those places' check-in instances. Therefore, places with variable degree of check-in frequencies were presented in each sample dataset. In addition, with this sampling technique the ratio of the number of samples to the number of unique places remained close to the original training dataset.

In sampling for the Random Forest algorithm, increasing the sample size does not necessarily lead to a better model, although the performance may improve. In contrast, the improvement in the model performance can be a result of overfitting. Increasing the sample size decreases the randomness of samples, and as a result, the Random Forest trees will be more similar to each other and the model tends to overfit. Additionally, with a large number of samples or number of classes, *randomForest* R package encounters execution halt. To explain the cause of this issue consider *nSample* as the number of dataset samples, and *nClass* as the number of classes. In the *randomForest* algorithm structure, the value of (*nSample* $\times$ *nClass*) is stored in an integer variable; therefore, if this value becomes very large, the algorithm execution will be halted due to integer overflow error. Table II represents the trained classifiers on drawn samples from the original dataset along with their performance and training details. The outcome of the trained models indicated a considerable improvement in model performance after applying the feature noise handling technique presented in previous section. Most noise handling mechanisms are focused on class noise. However, the studied check-in dataset contains a high level of feature noise. The polluted features are the location coordinates. Therefore, most noise handling mechanisms are not effective on our check-in dataset.

We also ran a *ranger* implementation of the Random Forest on one of the samples and compared its runtime and performance with parallel *randomForest*. Table III compares the results of the two packages of Random Forest implementations. As it is presented in the table, parallel *randomForest* provides better performance than ranger with a considerably shorter runtime.

VI. Conclusion

To summarize the main contribution of this work, the goal of reducing feature noise impacts on the machine learning algorithm training is achieved. The proposed technique fits into the datasets with the location coordinates representing the physical location; therefore, it is applicable on other datasets with location feature noise. A noise handling technique is essential in the check-in big datasets with high level of noise. Instead of employing other noise handling mechanisms, we designed and developed our preprocessing feature noise handling approach to correct feature noise instances without eliminating them from the dataset. Handling the features

TABLE II
EVALUATION OF THE RANDOM FOREST CLASSIFIERS. 16 CPU CORESARE UTILIZED. NCLASS IS THE NUMBER OF UNIQUE PLACES IN EACH SAMPLE.

Samples	**Sample-1**	**Sample-2**	**Sample-3**
Sample Size Train Size Test Size nclass	174,181 121,927 52,254 943	250,778 175,545 75,233 1257	274,206 191,944 82,262 1,357
sample size to the no. of unique places ratio	184.70	199.50	202.06
R package	Random Forest	Random Forest	Random Forest
Training Time (in minute)	12	24	27
MSE Before Cleansing	13.49%	13.11%	14.24%
MSE After Cleansing	4.93%	4.85%	4.85%
Correctly Predicted before Cleansing	86.50%	86.88%	85.76%
Correctly Predicted After Cleansing	95.06%	95.15%	95.14%

TABLE III
R PARALLEL *randomForest* PACKAGE VS. *ranger* PACKAGE. *nClass* IS CLASS NUMBER IN EACH SAMPLE.8 CPU CORES ARE UTILIZED.

Sample size: 274,206
Train Size: 191,944
Test Size: 82,262
nClass: 1,357

R package	Parallel *randomForest*	*ranger*
Training Time	27 minutes	3.5 hours
MSE	4.85%	6.31%
Correctly Predicted	95.14%	93.68%

noise significantly enhanced the classifier's learning process and accordingly enhanced the proposed Random Forest model performance.

In training Random Forest model, the parallel R *randomForest* package generated the classifier faster and more accurate than R *ranger* package. Although according to [15] the *ranger* package has better runtime and performance for high-dimensional datasets, it is not as effective as parallel *randomForest* for datasets with large number of class levels.

The represented method can gradually move to deeper levels in future works. For example, the value of erroneous coordinates can be replaced with a more representative value instead of the mean, or the borders of places can be defined utilizing a density-based classifier.

Finally, we draw the following conclusions for interested readers to deal with noise: 1) Determine the type of noise: class noise or feature noise 2) For feature noise try to design your own handling mechanism by determining the cause of noise appearance 3) Be aware of the over cleansing impacts and indicate a reasonable cleansing threshold (guideline).

ACKNOWLEDGMENT

The authors would like to thank Dr. Soroosh Sohangir for his thoughtful guidance and Tom Imboden for his generous sharing of computing resources to accomplish this study.
This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1053575. This effort was also supported by Southern Illinois University (SIU) which designated research funds.

REFERENCES

[1] M. Flintham, R. Anastasi, S. Benford, A. Drozd, J. Mathrick, D. Rowland, A. Oldroyd, J. Sutton, N. Tandavanitj, M. Adams *et al.*, "Uncle roy all around you: mixing games and theatre on the city streets." in *DiGRA Conference*, 2003.
[2] W. Broll, J. Ohlenburg, I. Lindt, I. Herbst, and A.-K. Braun, "Meeting technology challenges of pervasive augmented reality games," in *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*. ACM, 2006, p. 28.
[3] H. Gao and H. Liu, "Data analysis on location-based social networks," in *Mobile social networking*. Springer, 2014, pp. 165–194.
[4] A. Noulas, S. Scellato, N. Lathia, and C. Mascolo, "Mining user mobility features for next place prediction in location-based services," in *Data mining (ICDM), 2012 IEEE 12th international conference on*. IEEE, 2012, pp. 1038–1043.
[5] kaggle. Facebook v: Predicting check ins. [Online]. Available: http://www.kaggle.com/c/facebook-v-predicting-check-ins
[6] R. Y. Wang, V. C. Storey, and C. P. Firth, "A framework for analysis of data quality research," *IEEE transactions on knowledge and data engineering*, vol. 7, no. 4, pp. 623–640, 1995.
[7] R. Y. Wang and D. M. Strong, "Beyond accuracy: What data quality means to data consumers," *Journal of management information systems*, vol. 12, no. 4, pp. 5–33, 1996.
[8] B. Frénay and M. Verleysen, "Classification in the presence of label noise: a survey," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 5, pp. 845–869, 2014.
[9] X. Zhu and X. Wu, "Class noise vs. attribute noise: A quantitative study," *Artificial Intelligence Review*, vol. 22, no. 3, pp. 177–210, 2004.
[10] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
[11] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.
[12] P. Jeatrakul, K. W. Wong, and C. C. Fung, "Data cleaning for classification using misclassification analysis," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 14, no. 3, pp. 297–302, 2010.
[13] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Machine learning*, vol. 38, no. 3, pp. 257–286, 2000.
[14] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 2, no. 3, pp. 408–421, 1972.
[15] M. N. Wright and A. Ziegler, "ranger: A fast implementation of random forests for high dimensional data in c++ and r," *arXiv preprint arXiv:1508.04409*, 2015.