International Information and
Engineering Technology Association
Advancing the World of Information and Engineering

Enhancing GPS Accuracy with Machine Learning: A Comparative Analysis of Algorithms

Metin Zontul[1], Ziya Gokalp Ersan[2], Ilkay Yelmen[3*], Taner Cevik[4], Ferzat Anka[5], Kevser Gesoglu[6]

[1] Department of Computer Engineering, Faculty of Engineering and Natural Sciences, Sivas University of Science and Technology, Sivas 58000, Türkiye
[2] Department of Software Engineering, Faculty of Engineering and Architecture, Istanbul Gelisim University, Istanbul 34310, Türkiye
[3] Department of Computer Engineering, Faculty of Engineering and Natural Sciences, Istinye University, Istanbul 34396, Türkiye
[4] Department of Computer Engineering, Faculty of Engineering, Istanbul Arel University, Istanbul 34537, Türkiye
[5] Data Science Application and Research Center (VEBIM), Fatih Sultan Mehmet Vakif University, Istanbul 34445, Türkiye
[6] R&D Center, Turkcell Technology, Istanbul 34854, Türkiye

Corresponding Author Email: ilkay.yelmen@istinye.edu.tr

ABSTRACT

In the realm of wireless communications, the Global Positioning System (GPS), integral to Global Navigation Satellite Systems (GNSS), finds extensive applications ranging from vehicle navigation to military operations, aircraft tracking, and Geographic Information Systems (GIS). The reliability of GPS is often compromised by errors particularly prevalent in dense and structurally complex environments, where signal attenuation by environmental obstacles like mountains and buildings is common. These challenges necessitate the deployment of high-cost, precision GPS receivers capable of enhanced signal tracking and acquisition. This study investigates the reduction of GPS positioning errors by implementing a machine learning framework, utilizing a dataset from vehicle tracking devices equipped with Novatel and Ublox technologies. Ten machine learning prediction algorithms were evaluated, focusing on techniques that introduce randomness for stability, employ proximity for predictions, incorporate regularization to prevent overfitting, and leverage both single and ensemble methods to refine analyses. Among the evaluated algorithms, the Extra Trees algorithm was distinguished by its superior performance, achieving a coefficient of determination (R^2) of 99.6%, with the lowest error rates compared to its counterparts. The errors were quantified as Root Mean Square Error (RMSE) at 1.01E-4, Mean Absolute Error (MAE) at 4.14E-5, and Mean Square Error (MSE) at 1.03E+0 for normalized data. A comparative assessment across ten scenarios demonstrated that the machine learning-enhanced approach deviated by approximately 6.8 meters on average, markedly improving accuracy over traditional GPS methods and reducing positional deviations to a scale of meters. This advance represents a significant stride towards minimizing GPS inaccuracies in complex environments, providing a robust framework for enhancing navigational precision in critical applications.

1. INTRODUCTION

Recently, there has been increasing interest in the GNNS [1] framework in which positioning and geolocation processes are carried out. At least three satellites must be used to measure location in the GPS. The data acquired from multiple satellites is then processed and transformed into a single point [1]. GPS data is used in many fields, such as the defense industry, military, navigation, geology, and mapping. Although the measured GPS data sometimes differs from the actual location [2-4], fatal errors can occur due to the complexity of information, communication, and sensing technologies. Processing and converting GPS raw data into the target geolocation data is a complex and disciplined process requiring a wide range of information [4]. The most important argument is the value of GPS data accuracy. While it is

relatively easy to deal with the weak signal problem, accurate location determination is difficult as the data can give inaccurate results due to the hardware and technical shortcomings of GPS receivers. Although ground truth GPS receivers have higher precision than ordinary GPS receivers, they are very expensive. Therefore, while matching GPS point locations to the road network, improving the quality of the quality of ordinary GPS data based on ground truth GPS data is often required.

In general, GPS receivers learn the location of four or more satellites and find the distance to each. According to the distance to the satellites, it extracts the coordinate information of its location. This process relies on a basic mathematical operation known as trilateration. Using the trilateration method, the location of the object in three-dimensional space is revealed. The trilateration method is an approach for finding

a coordinate whose distance from two or more points is known (Figure 1). Circles are drawn with known points as centers and known distances as radii. By intersecting these circles, the desired coordinate can be found [5]. This is also the approach inspired by our method.
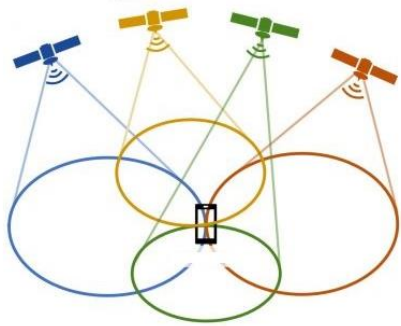
Figure 1. GPS trilateration calculation

1.1 Motivation

Most map-matching algorithms connect GPS tracks with the road network, considering the location of each GPS point or other GPS point in the same orbit [6, 7]. However, the map-matching algorithm does not consider the sources of error associated with the navigation sensors and the digital maps when estimating the location of the vehicle in the identified road segment. Indeed, the measured GPS data may deviate from the actual position. The basic mapping techniques are based on geometric analysis (point-to-point, point-to-curve, and curve-to-curve, using the geometric information of the digital road network) and topological analysis. Some basic map-matching algorithms try to overcome the errors with more complex techniques such as fuzzy logic, Kalman filtering, particle filtering, Bayes filtering, and Dempster. No matter how efficient these techniques are, the complexity of the street network can result in poor results.

In summary, geometric calculations in map matching and location estimation algorithms cannot give precise results when the complexity of the road network is high and the number of sample data points is low. As an alternative to geometric methods, more advanced map-matching algorithms such as particle filtering and Kalman filtering can be used. However, the particle filter algorithm needs a large number of particles to produce accurate results. The calculation costs are high. At the same time, there is no convenient way of relating particle counts to accuracy. For the same input, a particle filter can produce different outputs, making prediction success and error rates difficult to eliminate. On the other hand, the Kalman filter is used for linear problems. The Kalman filter can perform computation by iteratively updating linear Gaussian systems. However, it cannot properly model systems with nonlinear properties. Machine learning mechanisms can offer a number of solutions to address the shortcomings of sensor fusion-based approaches such as particle and Kalman filters. However, since the studies in this field have to use the whole data set, costly operations are carried out [8, 9].

1.2 Contribution

The detection of patterns in GPS error has been emphasized in many studies [10-14]. The position is calculated by the GPS receiver based on the distance to visible satellites. However,

these distances are distorted by errors. In this paper, Machine Learning (ML) algorithms are used to measure the accuracy of GPS data obtained from vehicle tracking devices to match real-time road details. As mentioned earlier, the ML approach is suitable due to some shortcomings of existing solutions. In this study, we use 10 different ML algorithms in our method to minimize the possible errors of GPS systems, especially in dense obstacles and complex areas. Since there are numerical data such as latitude, longitude, and time in the data, the use of these algorithms provides a serious advantage. However, in this study, we only give the latitude and longitude information of the data set as an input to ML algorithms with a unique approach, thanks to the model we propose. The program automatically generates feature values with algorithms (haversine, Manhattan, bearing) that calculate K-means and distance. Then, machine learning algorithms are used with these feature values. Therefore, transactions will be both fast and cost-effective. This process is described in Section 3.

On the other hand, the Kalman filter is applied to the integration of sensors and data fusion. In real-time systems, a series of sequential measurements is commonly utilized rather than a single observation to determine the system's state. For every time increment, this filter provides predictions of the actual unknown quantities and assesses their uncertainty. In the previous version of this study, map mapping was performed using Kalman Filter, and it was seen that Kalman Filter gave good results in location data with noisy data type as shown in experimental results below.

Therewithal, as an alternative approach to correcting GPS points and reducing device deviation rates in vehicle tracking systems, this study focuses on an extra tree-based method to measure the accuracy of matching GPS data obtained from vehicle tracking devices with real-time road details. GPS data is equipped with different regression algorithms, and the predictions are compared with the real data set. In addition, costly and expensive GNSS devices are used to obtain location data with high accuracy along the road route. In this study, a model using machine learning algorithms is presented as an alternative solution to advanced and expensive GNSS devices. By training the raw location GPS data (location data not on the road route) with the model, the accuracy of the latitude and longitude values that should be on the road line has been increased. This model can also be used efficiently in many new trending technologies, such as the Internet of Things [15], smart networks [16], and autonomous vehicles [17].

The rest of the paper is organized as follows: Section 2 gives a brief overview of several works related to GNSS in the literature. Section 3 describes the developed models, and section 4 presents the results of the experiments conducted and discussions among them. Finally, Section 5 includes the conclusions.

2. RELATED WORKS

Most map matching and location estimation algorithms are based on the theory of alignment with the current road network and the measurement of other GPS points in the same orbit with respect to the location of each GPS point [18]. Although these geometric calculations are accurate, they cannot produce conclusive results when the sampling rate is small or the complexity of the road network is high [6]. As an alternative to geometric approaches, more sophisticated map-matching algorithms such as particle filters and Kalman filters are used.

In applications that are resolved by a particle filter, the algorithm calculates probable values based on the candidate paths in orbit and the GPS point values [19]. The most reliable route between these results is used to suit the map. Particle filter technology is used in many applications, such as path tracking and prediction in non-linear dynamic systems [20]. However, the particle filter algorithm requires a large number of particles to produce accurate results; thus, the cost of the calculation appears to be high. Moreover, there is no convenient way to correlate accuracy with particle numbers. For example, a particle filter for the same input can produce different outputs, making predictable success and error rate elimination difficult [21]. A particle filter is formulated as follows:

$$\pi_n(dx_n \,|y_{1:n}) \approx \hat{\pi}_n(dx_n \,|y_{1:n})$$
$$= \sum_{i=1}^{N} W_n^i \, \Delta_{X_n^i}(dx_n) \qquad (1)$$

where, W_n^i collects random weights, while X_n^i are the random variables called snippets, Δ_x is the mass of points at zero time, π_0 pulls particles from $W_0^i = 1/N$ which is set in n time $\hat{\pi}_{n-1}$ that starts as new particles $P(.\,|X_n^i)$, from which X_n^i is drawn. In Eq. (2), W_{n-1}^i having weights X_n^i particles $\pi_{n|n-1}$ provides a more important approach. $W_n^i \propto W_{n-1}^i \, g(\,y_n\| \, X_n^i)$ weights are updated, and iteration is closed.

$$\pi_{n\,|\,n-1}(dx_n \,|y_{1:n-1})$$
$$= \int \pi_{n-1}(dx_{n-1} \,|y_{1:n-1})P(dx_n \,|x_{n-1}) \qquad (2)$$

$$\pi_n(dx_n \,|y_{1:n})$$
$$= \pi_{n\,|\,n-1}(dx_n \,|y_{1:n-1}) \frac{g(y_n|x_n)}{p_n(y_n|y_{1:n-1})} \qquad (3)$$

Another method, the Kalman filter, is used to maximize the interest variables when they are not specifically measured. In addition, it is both reciprocal and online, making it ideal for real-time signals. Filters from the Kalman linear system are popular due to their minimum average square estimate. Furthermore, the Kalman filter can recurrently update Gaussian linear systems and perform the calculation process. However, systems with non-linear properties cannot be properly modeled [22]. The Kalman filter is developed and motivated as an ideal filter for linear systems. For targets with variable speeds or accelerations, Kalman and particle filters are the only reliable ones [23]. The extended Kalman filter

provides solutions to nonlinear evaluation problems by applying standard Kalman filter formulas after a linearization process. Nevertheless, filter operations cause approximation errors in linearization and updating operations. The Kalman filter is formulated in the simplest form as follows:

$$\hat{x}_k = K_k.Z_k + (1 - K_k).\hat{x}_{k-1} \qquad (4)$$

where, k indicates the states. The goal is to find $\hat{x}_k$ value for each k. Z_k is the measured value, while K_k is the most important parameter of Kalman gain. Finally, $\hat{x}_{k-1}$ is the prediction of the previous state of the signal [24, 25].

In addition to approaches such as particle and Kalman filters, various solutions are also offered on the machine learning side. A model based on the estimation of GPS data is given in Markovic's study [26] to calculate vehicle travel times in the area. For an accurate assessment and road network, the K-Nearest Neighbor (KNN) algorithm is utilized to map GPS data. However, as revealed in this study, in contrast to the different ML algorithms, the KNN algorithm does not always yield accurate results. As can be understood from here, systems based on a single algorithm cannot produce consistent results under all conditions. In another analysis on ML [27], the exactingness of the GPS points along the way is achieved by training the models of the Artificial Neural Network (ANN). However, the model using reverse artificial neural networks has been found to slow down the training period.

3. METHODOLOGY

In this study, we investigated ten different ML techniques, such as Extra Trees, KNN, Linear Regression, Ridge, Lasso, Elastic Net, Random Forest, Decision Trees, Gradient Boosting, and Ada Boosting, to reduce the error deviation rates of GPS data. The "minimum error rate" refers to the lowest achievable misclassification rate for our predictive models. This metric is pivotal in evaluating the performance and robustness of our chosen methods against the complexity and variability inherent in the data set. Since numerical data such as latitude, longitude, and time are present in the data, it is acceptable to use these algorithms in research. The outputs of the regression model also include the error in magnitude and direction. In the working mechanism of the proposed method, as shown in Figure 2, the first step is to prepare the dataset. Novatel and Ublox GNSS devices are used while preparing the raw dataset [28]. Since Ublox can make more precise measurements, it is used as ground truth (reallat, reallong) data.
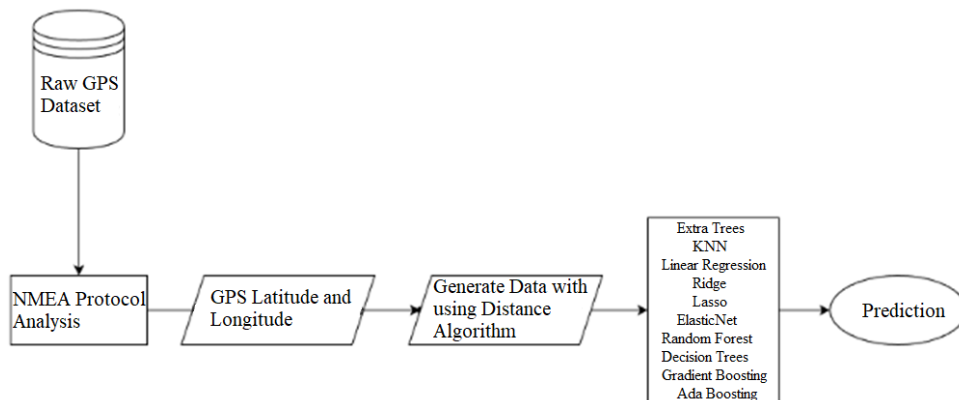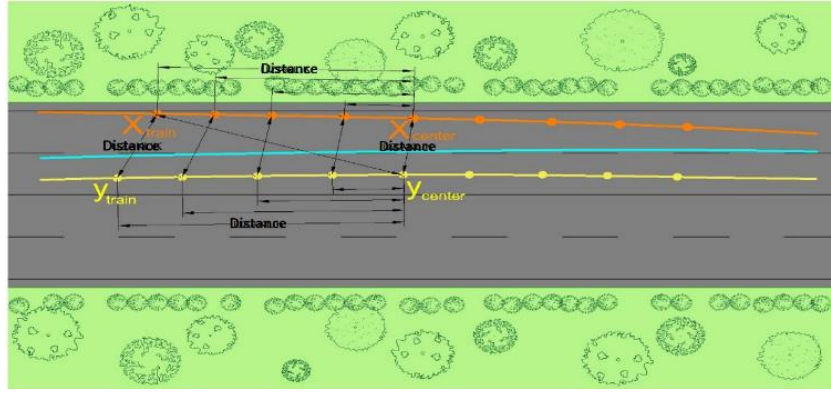

Figure 2. A schematic view of all phases of the proposed methodology

Figure 3. The working mechanism of the proposed method in the data processing

Using haversine, Manhattan, and bearing distance algorithms on raw latitude, longitude data from a Novatel GNSS device, and real point data from Ublox, our input data became 22 features in total. Then, the data set is reduced to 17 features by applying Principal Component Analysis (PCA) analysis.

Figure 3 shows how we get each feature. The dots in orange here are the raw latitude and longitude values. The points found as centers are calculated with the K-means algorithm. Yellow dots are ground truth (reallat, reallong) values. The prediction data from our machine learning-based method is shown with a blue line.

The rest of this section refers to the regression models configured specifically for this study and whose mathematical logic is described in the basic machine learning literature [29].

3.1 Extra trees (extremely randomized trees)

Following the traditional top-down procedure, the Extra Trees algorithm creates several regression tree assemblies. There are two key differences from other tree-based grouping approaches: the separation of nodes by randomly selecting split points and the use of the entire learning instance (rather than a boot-copy) to grow trees. Applying randomization and using the entire original sample contribute to the reduction in variance and bias, respectively. Although the Extra Trees algorithm applies a similar procedure as the other tree-based algorithms, it is faster than their counterparts. Algorithm 1 shows the splitting procedure in Extra Trees for numerical properties [30].

Algorithm 1. Extra-Trees splitting algorithm (for numerical attributes)

Split_a_node(S)
Input: the local subset S corresponding to the node we want to split
Output: a split $[a<a_c]$ or nothing.
If **Stop_split**(S) is TRUE then return nothing.
Otherwise select K attributes $\{a_1,.....,a_K\}$ among all non constant (in S) candidate attributes;
Draw K splits $\{s_1,.....,s_K\}$, where $s_i =$ **Pick_a_random_split**(S, a_i), $\forall i = 1,......,K$;
Return a asplit s_* such that Scores(s_*, S) = $\max_{i = 1,.....,K}$ Score(S_i, S).

Pick_a_random_split(S,a)
Inputs: a subset S and an attribute a
Output: a split
Let a_{max}^S and a_{min}^S denote the maximal and minimal value of a in S;
Draw a random cut-point a_c uniformly in a_{min}^S, a_{max}^S;
Return the split $[a<a_c]$.

Stop_split(S)
Input: a subset S
Output: a Boolean
If $|S| < n_{min}$, then return TRUE;
If all attributes are constant in S, then return TRUE;
If the output is constant in S, then return TRUE;
Otherwise, return FALSE.

3.2 KNN

The KNN algorithm, presented in the literature as a lazy learning method, was developed by Cover and Hart [31]. The KNN algorithm is a simple algorithm that reclassifies all existing states according to a similarity measure (e.g., distance functions such as Euclidean and Manhattan, as given in Eq. (5, 6) [24, 25].

$$dEuclidean = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2} \qquad (5)$$

$$dManhattan = \sum_{i=1}^{k}|x_i - y_i| \qquad (6)$$

It looks at the distance of the data to be classified with its neighbours and performs the classification and regression process with the most appropriate label. In the proposed framework, the property value of the object is used as output in KNN regression algorithms. This is the average of the nearest neighbour values. Since the inputs are decimal values (e.g., latitude and longitude), they are assumed to be similar to these values in this analysis. Therefore, the regression property of the KNN algorithm is used, which is not graded.

3.3 Linear regression

Linear regression [32, 33] is the most common predictive model that describes the relationship between variables. As well as univariate or multivariate data types, the concept is linear. There are two different linear regression models: simple linear and multiple linear regression. Linear regression is described in Eq. (7).

$$y = x\beta + \varepsilon \qquad (7)$$

In Eq. (7), x is a dependent variable that is always a continuous value and y is the independent variable that can be either a continuous or categorical value. Its analysis is done with a probability distribution and focuses on multivariate analysis with a conditional probability distribution.

3.4 Ridge regression, lasso, and elastic net

Ridge regression [34] is suitable if it has many predictors and all non-zero coefficients from a normal distribution. In particular, it performs well with many predictors, each with a small impact, and prevents poor determination of the coefficients of linear regression models with many associated variables and high variance. Ridge regression equally shrinks the coefficients of associated predictors towards zero. That is, for example, given k identical predictions, each would obtain identical coefficients equal to the dimension any predictor would take if they fit alone [35]. Ridge Regression therefore does not force the coefficients to disappear, and therefore we cannot choose a model that only has the most relevant and predictive subset of predictors.

$$\hat{\beta}(ridge) = arg_{\beta} min \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2 \qquad (8)$$

In Eq. (8), β is the regression coefficient; the higher the value of λ, the greater the fractures in the data. Since the value of λ is data-dependent, data-based methods such as cross-validation can be used [36].

Lasso regression methods are widely used in areas with large data sets, such as genomics, where fast and efficient algorithms are required. Whereas Lasso regression is not robust against high correlations between predictors. It arbitrarily chooses one, ignores the others, and collapses when all predictors are the same [33]. The lasso penalty expects only a small subset to be larger (but not zero) and many coefficients to be close to zero.

$$\hat{\beta}(lasso) = arg_{\beta} min \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \qquad (9)$$

In Eq. (9), β defines the regression coefficient, $\lambda \geq 0$ is set. Reducing the lassoed components by the least-squares method by reducing the best-selected λ values to zero [37].

The Elastic Net (ENET), a combination of Ridge and Lasso, is robust to extreme correlations between the predictors [38, 39]. ENET has been proposed to analyze high-dimensional data to avoid the instability of lasso solutions when predictors are highly correlated. The ENET uses a mixture of the Ridge and Lasso penalties and can be expressed as in Eq. (10) [40].

$$\hat{\beta}(enet) = \left(1 + \frac{\lambda_2}{n}\right)\{arg_{\beta} min \|y - X\beta\|_2^2 + \lambda_2 \|\beta\|_2^2 + \lambda_1 \|\beta\|_1\} \qquad (10)$$

In this study, ridge, Lasso, and Elastic Net algorithms are used subsequently to avoid overestimation and estimation of new points on normalized data.

3.5 Random forest

Random Forest (RF) is an ensemble learning method that represents an important advance in machine learning. The task of the RF method is to create a large number of individual decision trees during the training phase. Its aim is to find new ways to combine information from individual trees [41].

$$H_{regress}(x) = \frac{1}{T}\sum_{t=1}^{T} h_t(x,\theta) \qquad (11)$$

In Eq. (11) T shows the number of forest trees. A random

vector of h_t is used to decide how successive cuttings are produced in the development of individual trees. It is assumed that x and the samples are trained [42] independently. In this study, the tree depth is partitioned into five layers to calculate the GPS points more accurately.

3.6 Decision trees

In decision trees, the tree structure consists of a root node, internal nodes (branches), and end nodes (leaves). The tree structure is configured according to the training data. The attributes of the training samples are used to determine the placement of the nodes in the tree and threshold coefficients. The created decision tree reaches the end node by moving at the nodes according to the attribute values and threshold values determined during the training process. Optimal threshold values should be determined to make the branches leaving the nodes have the highest difference or to minimize the similarity of the crossed nodes [43].

$$\hat{y}_t = \frac{1}{N_t}\sum_{i \in D_t} y^{(i)} \qquad (12)$$

In Eq. (12), N_t is the number of trained samples in node t, D_t is the training subset, $y^{(i)}$ is the actual targeted value, and $\hat{y}_t$ is the estimated value [44]. In this study, since GPS point estimation is a value estimation problem, a decision tree structure based on regression is formed, which takes less training time.

3.7 Gradient boosting decision trees

Gradient Boosting Decision Trees (GBDT) [45] is an additional regression model that includes a set of decision trees. A single decision tree has an overfitting problem, but the GBDT algorithm can overcome this by combining hundreds of weak decision trees consisting of several leaf nodes. GBDT has several advantages, including the ability to find nonlinear transformations, the ability to handle skewed variables without the need for transformations, computational robustness, and high scalability [46].

$$c_{tj} = argmin \sum_{x_i \in R_{tj}} L(y_i, f_{t-1}(x_i) + c) \qquad (13)$$

where, R_{tj} is the number of leaf nodes in the tree, f_{t-1} is the most accurate value from the previous iteration, and c_{tj} represents the best output value for each leaf node [47].

4. EXPERIMENTAL RESULTS

The main idea of location algorithms is to model regular patterns based on historical information and to determine the most probable movements based on current observations. While a great deal of attention is paid to network architecture design literature in order to endorse GPS generally, nothing is done on the essential issue of the appropriate location. Besides, we concentrate instead of costly equipment on estimating the target position using ML techniques.

The trajectory of a vehicle cannot be explicitly converted to an approximation model because of changes in position and time. The trajectories of the vehicle do not exactly follow the same route every day, but they do not. It may result in higher

calculation costs, as well as a poorer estimate of a location with continuous coordinates [48]. The relationship between latitude and longitude coordinates along the trajectory must be defined before predictive models are developed.

4.1 Dataset materials

In this study, we meticulously collected vehicle location data, including latitude and longitude information, across five distinct districts in Istanbul, Türkiye—Besiktas, Ortakoy, Palanga, Buyukdere, and Eski Buyukdere. This dataset represents an accumulation of data over four days, conducted in two separate laps for each day to ensure consistency and reliability in our findings. The primary objective of our data collection was to analyze the precision and accuracy of two GNSS (Global Navigation Satellite System) receivers: Novatel and Ublox. The choice of these devices was guided by their widespread use in geographic data collection and their distinct operational capabilities. Novatel is the GNSS receiver platform for recording GPS data with multiple frequency tracking [49]. Ublox is a GNSS receiver, just like Novatel. However, Ublox can make more accurate measurements [50]. The latitude and longitude data obtained through Novatel is 6027, while the dataset from Ublox is 1407 points. The latitude and longitude points taken with Ublox represent more sensitive data. Therefore, Ublox is used as the ground truth data in our dataset. The raw data gathered from GNSS receivers is received using the National Marine Electronics Association (NMEA) protocol. NMEA is a protocol created by the US National Maritime Electronics Association. Sentences begin with a $ sign, and the GP letter is then added to indicate that it is from the GPS receiver. The next three letters indicate the purpose of the sentence (e.g., $ GPVTG: velocity, $ GPGLL: position). What is most important here is the GGA (Global Positioning System Fix Data). "Fix" is a GPS term referring to the determination of the current location [51]. An NMEA example sentence is given in the following:

$GPGGA,073840.00,4102.7502506, N,02901.1513588, E,2,12,1.0,9.979, M,37.30, M,18, TSTR*5

In the example sentence, the part up to N (4102.7502506, N) represents latitude, the part up to E (02901.1513588, E) represents longitude, and the time in UTC format is provided first (073840.00). All the raw data (6027 dots) in NMEA format is coded in Python and appropriately divided into latitude, longitude, and time zones. With the new dataset, test and training data are created in advance. The generated training data is modeled with regression algorithms in ML.

4.2 Accuracy performance analysis

Data are shown on the Open Street Map and Google Map as a result of the analysis. The correlation coefficient is determined by extracting the error rates. The absolute values of the discrepancies between the test and forecast values can be seen as the MAE. All errors are of equal weight in MAE [52]. The MAE formula is given as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \qquad (14)$$

where, n is observed data, y_i is test data, and $\hat{y}_i$ is the estimated data.

Root Mean Square Error is used to measure the difference between the predicted values, similar to MAE. The difference is that it can calculate larger absolute values by giving more weight than the MAE gives. The greater the difference between MAE and RMSE, the greater the error rates are in the sample [53].

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \qquad (15)$$

where, n is observed data, y_i is test data, and $\hat{y}_i$ is the estimated data. As the input data of the algorithm, the methods used in distance calculations (Haversine, Manhattan, Bearing), the K-means clustering algorithm, PCA, reduced and simplified data, vehicle speed, and real road information are processed. The theorem of Haversine is used to measure the latitude and longitude values of a two-point range on the surface of the Earth. To calculate two distances, four variables must be prepared. It is estimated that the distance between the two points is 6 367.45 kilometers [54]. The Haversine formula is as follows [35]:

$$a = sin^2(\Delta\lambda/2) + cos\,\varphi1 \cdot cos\,\varphi2$$
$$\cdot sin^2(\Delta\lambda/2)$$
$$c = 2 \cdot atan2(\sqrt{a}, \sqrt{(1-a)})$$
$$d = R \cdot c \qquad (16)$$

where, ϕ is the latitude, λ denotes longitude, and R stands for the radius of the world. The calculation of distance by Manhattan (city block size) calculates the size to travel on a grid-like path from one data point to another. The distance between the two elements in Manhattan is the sum of the differences between the components. The Manhattan distance is also known as the *L1* distance [55]. The Manhattan distance is formulized as follows:

$$d = \sum_{i=1}^{n}|x_i - y_i| \qquad (17)$$

where, n is the number of the variables, x and y represent the two points.

The bearing algorithm, on the other hand, is the calculation of the distance between two points on the earth's surface by connecting them using a circular line. The bearing distance formula is as follows [41]:

$$\theta = atan2(sin\,\Delta\lambda \cdot cos\,\varphi2 , cos\,\varphi1 \cdot sin\,\varphi2$$
$$- sin\,\varphi1 \cdot cos\,\varphi2 \cdot cos\,\Delta\lambda\,) \qquad (18)$$

where, ϕ_1, λ_1 are the starting points, ϕ_2, λ_2 refer to the ending points, and $\Delta\lambda$ denotes the longitude difference.

The K-means clustering algorithm allows grouping of data that show similar properties in a dataset. The K value in the algorithm sets the number of clusters, and this value is to be taken as a parameter [56].

$$J = \sum_{j=1}^{k}\sum_{i=1}^{n}\left\|x_i^{(j)} - c_j\right\| \qquad (19)$$

where, J is the target function, k refers to the number of clusters, n denotes the number of cases, $x_i^{(j)} - c_j$ is the distance function, and c_j is the centroid for cluster j [57].

Table 1. Procedures applied to the input and output data

| Input	Description
Lat	Latitude (Novatel)
Lon	Longitude (Novatel)
Time	GPS time
Haversine_latlon_reallatlon	Distance between Latitude, Longitude and Real Latitude, Real Longitude according to Haversine
Manhatten_latlon_reallatlon	Distance between Latitude, Longitude and Real Latitude, Real Longitude according to Manhatten
Bearing_latlon_reallatlon	Distance between Latitude, Longitude and Real Latitude, Real Longitude according to Bearing
Centroid_long	The centroid of Longitude points
Centroid_lat	Centroid of Latitude points
Centroid_real_long	The centroid of Real Longitude points
Centroid_real_lat	The centroid of Real Latitude points
Haversine_latlon_centlatlon	Distance between Latitudes, Longitudes, and Centroid to Haversine
Haversine_reallatlon_centreallatlon	Distance between Real Latitudes, Real Longitudes, and their Centroids according to Haversine
Haversine_centlatlon_centreallatlon	Distance between Real Latitudes, Real Longitudes
Haversine_latlon_centreallatlon	Distance between Latitudes, Longitudes, and Centroids Real Latitudes, Real Longitudes according to Haversine
Manhatten_latlon_centlatlon	Distance between Latitudes, Longitudes, and Centroid according to Manhattan
Manhatten_reallatlon_centreallatlon	Distance between Real Latitudes, Real Longitudes, and their Centroids according to Manhattan
Manhetten_centretlon_centrealletlon	Distance between Real Latitudes, Real Longitudes
Bearing_latlon_centletlon	Distance between Latitudes, Longitudes, and Centroid according to Bearing
Bearing_reallatlon_centlatlon	Distance between Real Latitudes, Real Longitudes, and Centroids Latitudes, Longitudes according to Bearing
Bearing_centlatlon_centreallatlon	Distance between Real Latitudes, Real Longitudes
Speed_haversin	Speed between two points according to the distance calculated by Haversine
Speed_manhatten	Speed between two points according to the distance calculated by Manhattan
Reallot	Real Latitude – output value
Reallon	Real Longitude – output value

Table 2. Estimation error rates

Our Method Based on ML Algorithms	R^2	RMSE	MAE	MSE
Extra Trees	0.996139756	0.000101322	0.0000414464	1.026606022
KNN	0.946238279	0.000377134	0.0000812606	1.422298635
Linear Regression	0.987449608	0.000182063	0.000100999	3.314678968
Ridge	0.604053192	0.00104511	0.0005396492	1.092254988
Lasso	0.467520898	0.001701065	0.000824387	2.89362048
Elastic Net	0.517820798	0.001701065	0.000824387	2.89362048
Random Forest	0.970729381	0.000297608	0.000100891	8.857064416
Decision Trees	0.979705547	0.000240813	0.0000776682	5.799076299
Gradient Boosting	0.569920497	0.001115341	0.00052906	1.243984473
Ada Boosting	0.897450938	0.000699161	0.000193861	4.888266597

The distances between the latitude and longitude values that are retrieved from the GPS are calculated for the exact spot. The midpoints of the latitudes and longitudes are calculated using the equation K-means. A total of 22 data are generated as inputs, and 2 data are generated as outputs. Table 1 shows the procedures applied to both input and output data throughout the proposed methodology.

Table 2 shows the results of ten different ML algorithms that are compared. The Extra Trees algorithm yielded 99.6% accuracy in R^2 with low error values. The errors RMSE, MAE and MSE are calculated in terms of normalized values. Therefore, they are expected to be very close to zero and each other in optimal regression models.

MAE measures the average magnitude of errors in a series of estimates, which represents the mean of the validation sample of the absolute values of the differences between the estimate and the observed. RMSE is a second-order scoring rule that measures average error magnitude. The difference between the estimate and the observed values is taken from each frame and then centered on the sample. Both evaluation tools can be used to diagnose the variation of errors in several estimates. GPS latitude and longitude estimation results obtained from test data are shown on the map.

According to the results obtained, the method based on the Extra Tree Algorithm performs better. The working results of this are shown on the sample real map in Figure 4. On the other hand, the latitude and longitude values gathered from the Novatel device are shown on the map in Figure 5 to help understand the difference between them.
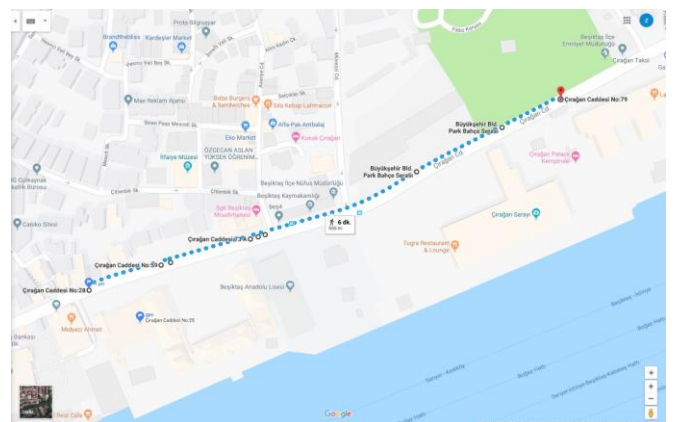
Figure 4. The extra trees algorithm results (real map of Istanbul)
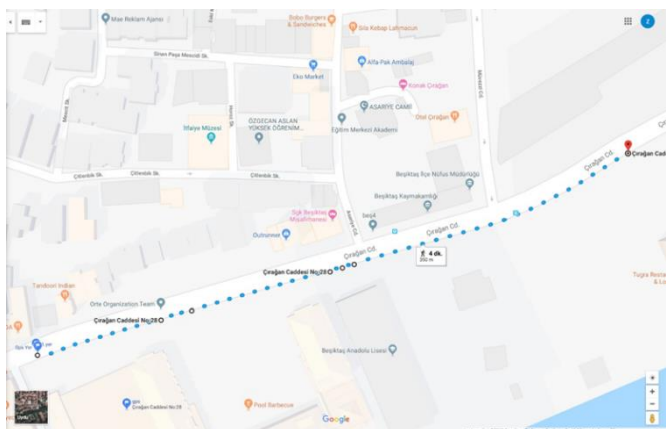
Figure 5. Latitude and longitude values gathered from the Novatel device (real map of Istanbul)

As clearly seen on the map, the coordinates retrieved from GPS (Figure 5) do not produce more consistent and accurate results along the road compared to the Extra Trees algorithm (Figure 4), which is one of the ML algorithms. Obviously, the Extra Trees algorithm correctly calculates the latitude, longitude points and draws a smoother path along the road. In addition, Table 3 expresses the deviation rates, which are calculated by using Euclid's theorem. The deviations from

exact locations with respect to the proposed prediction model are between 1.9 and 9.4 meters approximately. Therefore, the average of deviations is 6.8 meters with respect to exact locations based on ground truth GPS receivers. This finding marks a significant advancement, notably reducing the deviation to a scale of meters when comparing GPS data from the vehicle tracking system with the projected coordinates. Importantly, the deviation of 6.8 meters is evaluated within the context of its practical implications in various application areas, such as vehicle navigation and search and rescue operations. In vehicle navigation, such precision significantly enhances route accuracy and safety, making it highly beneficial for autonomous driving technologies where every meter counts. In search and rescue missions, this reduced deviation could drastically improve the efficiency of locating individuals in distress, potentially saving lives by enabling quicker response times. By benchmarking against acceptable standards within these application areas, it becomes evident that the achieved deviation substantially exceeds the conventional accuracy, thereby affirming the superiority and applicability of our method in real-world scenarios. In literature, the Kalman Filter mechanism, which was recently proposed [58], gave deviations between 5.2 and 27.2 meters with a 10.15 average value. In the analysis of the results obtained, the method proposed in this study demonstrates that it outperforms the others with the least deviation.

Table 3. Deviation results in meters

Proposed Method based on Extra Trees		Kalman Filter		GPS		Distance (Deviation Rate)
Latitude	Longitude	Latitude	Longitude	Latitude	Longitude	Meter
41.04268513	29.00984937	41.04410506	29.01486518	41.04267867	29.00981133	3.26992175
41.04284729	29.0103781	41.04414235	29.01494283	41.04283289	29.01039153	1.95796513
41.04307647	29.01123174	41.04417789	29.01502226	41.04300557	29.01121421	8.01926154
41.04310119	29.01134478	41.04421081	29.01510458	41.04305312	29.01141928	8.22238095
41.04330904	29.01228856	41.04424125	29.0151883	41.04324906	29.01234396	8.12768869
41.04332878	29.01238319	41.04427077	29.01527151	41.0432678	29.01242529	7.64459395
41.04334697	29.01246332	41.0443013	29.01535364	41.0432861	29.01250516	7.62372252
41.04390765	29.01425744	41.04373476	29.01394512	41.04384431	29.01433329	9.490165
41.04430288	29.01527054	41.04370114	29.01387296	41.04430417	29.01521512	4.64992499
41.04455555	29.01596397	41.04366641	29.01379503	41.04453592	29.01599378	3.31897928

5. CONCLUSIONS

This paper presents a study that aims to mitigate GPS localization errors by using ML methods. In the first step, the data set is created using the latitude and longitude coordinates obtained through Novatel (less precision) and Ublox (ground truth) GNSS receivers. This data set is analyzed within the framework. During the data set creation phase, the raw data in NMEA format is processed. After the data creation phase, location estimation and map matching operations are performed. Then, the latitude and longitude values calculated by machine learning-based methods are compared with the raw data set and shown numerically on the chart. As a result of the analysis, the false deviation rates of the GPS data are calculated and given in meters. It is observed that the Extra Trees algorithm outperforms the other 9 machine learning algorithms and the Kalman Filter method. New coordinate values are calculated with algorithms on Google Maps, and they are shown on the map with the help of OpenStreetMap. Compared with previous raw data, deviation rates are calculated in meters using the Euclidean principle. Consequently, machine learning algorithms can be used effectively to correct erroneous GPS points and reduce the

deviation rates of devices using navigation, especially for vehicle tracking systems. For future research, additional experiments at various locations using a variety of GNSS receivers are recommended to further highlight the value of machine learning algorithms in reducing GPS errors. Additionally, although our preliminary findings highlight a notable increase in positional accuracy (in particular, the Extra Trees algorithm outperforms nine other machine learning algorithms and the Kalman Filter approach), future research should include a broader range of datasets. Performing statistical significance analysis with a large data set will provide more robust validation of these results. This approach will help validate the effectiveness of machine learning algorithms in improving GPS accuracy on a more significant and diverse scale.

REFERENCES

[1] Li, D., Ma, X., Zhao, J., Wu., F. (2022). Mitigating GNSS multipath effects using XGBoost integrated classifier based on consistency checks. International Journal of

Antennas and Propagation, 1-14. https://doi.org/10.1155/2022/2742620

[2] Rychlicki, M., Kasprzyk, Z., Rosiński, A. (2020). Analysis of accuracy and reliability of different types of GPS receivers. Sensors, 20(22): 6498. https://doi.org/10.3390/s20226498

[3] Dogan, U., Uludag, M., Demir, D.O. (2014). Investigation of GPS positioning accuracy during the seasonal variation. Measurement, 53: 91-100. https://doi.org/10.1016/j.measurement.2014.03.034

[4] Nikolic M., Jovic J. (2017). Implementation of generic algorithm in map-matching model. Expert Systems with Applications, 72: 283-292. https://doi.org/10.1016/j.eswa.2016.10.061

[5] Orabi, M., Khalife, J., Abdallah, A.A., Kassas Z.M., Saab, S.S. (2020). A machine learning approach for GPS code phase estimation in multipath environments. In IEEE/ION Position, Location and Navigation Symposium (PLANS), Portland, OR, USA, pp. 1224-1229.
https://doi.org/10.1109/PLANS46316.2020.9110155

[6] Kempinska, K., Davies, T., Taylor, J.S. (2021). Probabilistic map-matching using particle filters. arXiv preprint arXiv:1611.09706.

[7] Hashemi, M., Karimi, H. (2016). A machine learning approach to improve the accuracy of GPS-based map-matching algorithms. In IEEE 17th International Conference on Information Reuse and Integration (IRI), Pittsburgh, PA, USA, pp. 77-86. https://doi.org/10.1109/IRI.2016.18

[8] Carron, A., Todescato, M., Carli, R., Schenato, L., Pillonetto, G. (2016). Machine learning meets Kalman filtering. In 2016 IEEE 55th Conference on Decision and Control (CDC), Las Vegas, NV, USA, pp. 4594-4599. https://doi.org/10.1109/CDC.2016.7798968

[9] Gao, X., Luo, H., Ning, B., Zhao, F., Bao, L., Gong, Y., Xiao, Y., Jiang, J. (2020). RL-AKF: An adaptive Kalman filter navigation algorithm based on reinforcement learning for ground vehicles. Remote Sensing, 12(11): 1704. https://doi.org/10.3390/rs12111704

[10] Kim, S., Byun, J., Park, K. (2022). Machine learning-based GPS multipath detection method using dual antennas. arXiv:2204, 14001. https://doi.org/10.48550/arXiv.2204.14001

[11] Hashemi, M. (2017). Reusability of the output of map-matching algorithms across space and time through machine learning. IEEE Transactions on Intelligent Transportation Systems, 18(11): 3017-3026. https://doi.org/10.1109/TITS.2017.2669085

[12] Lou, J., Cheng, A. (2020). Detecting pattern changes in individual travel behavior from vehicle GPS/GNSS data. Sensors, 20(8): 2295. https://doi.org/10.3390/s20082295

[13] Wang, Y., Qin, K., Chen, Y., Zhao, P. (2018). Detecting anomalous trajectories and behavior patterns using hierarchical clustering from taxi GPS data. ISPRS International Journal of Geo-Information. 7(1): 25. https://doi.org/10.3390/ijgi7010025

[14] Wang, C.Z., Kong, L.W., Jiang, J. (2021). Machine learning-based approach to GPS antijamming. GPS Solution, 25(115).

[15] Kiani, F., Nematzadehmiandoab, S., Seyyedabbasi, A. (2019). Designing a dynamic protocol for real-time industrial internet of things-based applications by efficient management of system resources. Advances in

Mechanical Engineering, 11: 1-23. https://doi.org/10.1177/1687814019866062

[16] Kiyani, F., Tahmasebi rad, H., Chalangari H., Yari, S. (2010). DCSE: A dynamic clustering for saving energy in wireless sensor network. In 2010 Second International Conference on Communication Software and Networks, Singapore, pp. 13-17. https://doi.org/10.1109/ICCSN.2010.98

[17] Kiani, F., Saraç, Ö.F. (2022). A novel intelligent traffic recovery model for emergency vehicles based on context-aware reinforcement learning. Information Sciences, 619: 1-22. https://doi.org/10.1016/j.ins.2022.11.057

[18] Islam, M.R., Kim, J.M. (2014). An effective approach to improving low-cost GPS positioning accuracy in real-time navigation. The Scientific World Journal, 2014: 671494. https://doi.org/10.1155/2014/671494

[19] Ahwiadi, M., Wilson, W. (2019). An enhanced mutated particle filter technique for system state estimation and battery life prediction. IEEE Transactions on Instrumentation and Measurement, 68(3): 923-935. https://doi.org/10.1109/tim.2018.2853900

[20] Xian, W., Long, B., Li, M., Wang, H. (2014). Prognostics of lithium-ion batteries based on the Verhulst model, particle swarm optimization and particle filter. IEEE Transactions on Instrumentation and Measurement, 63(1): 2-17. https://doi.org/10.1109/TIM.2013.2276473

[21] Pathak, A., Singh, E. (2014). Comparative study on filtering techniques of digital image processing. Advance in Electronic and Electric Engineering, 4(6): 669-674.

[22] Arulampalam, M.S., Maskell, S., Gordon, N., Clapp, T. (2012). A tutorial on particle filters for on-line non-linear/non-gaussian Bayesian tracking. IEEE Transactions on Signal Processing, 50(2): 174-188. http://doi.org/10.1109/78.978374

[23] Mahfouz, S., Mourad-Chehade, F., Honeine, P., Joumana, F., Snoussi, H. (2014). Target tracking using machine learning and Kalman filter in wireless sensor networks. IEEE Sensors Journal, 14(10): 3715-3725. https://doi.org/10.1109/JSEN.2014.2332098

[24] Nematzadeh, S., Kiani, F., Torkamanian-Afshar, M., Aydin, N. (2022). Tuning hyperparameters of machine learning algorithms and deep neural networks using metaheuristics: A bioinformatics study on biomedical and biological cases. Computational Biology and Chemistry, 97: 1-22. https://doi.org/10.1016/j.compbiolchem.2021.107619

[25] Çayiroglu, I. (2012). Kalman filter and programming. Science and Technology Information Sharing, 1: 1-6.

[26] Markovic, H., Basic, B., Gold, H., Dong, F., Hrvoje, G. (2010). GPS data-based non-parametric regression for predicting travel times in urban traffic networks. Promet-Traffic & Transportation, 22(1): 1-13. https://doi.org/10.7307/ptt.v22i1.159

[27] Han, Y., Kim, Y., Ku, J., Jung, Y., Roh, J. (2021). Map matching algorithm for real-time data processing of non-route GPS data in Seoul. KSCE Journal of Civil Engineering, 25(9): 3511-3522. http://doi.org/10.1007/s12205-021-1750-x

[28] Zhao, J., Hernández-Pajares, M., Li, Z., Wang, L., Yuan, H. (2020). High-rate Doppler-aided cycle slip detection and repair method for low-cost single-frequency receivers. GPS Solutions, 24(3). https://doi.org/10.1007/s10291-020-00993-0

[29] Theodoridis, S., Koutroumbas, K. (2008). Pattern Recognition, 4th Edition. Elsevier, Amsterdam, Netherlands, 23-59.

[30] Guerts, P., Ernst, D., Wehenkel, L. (2016). Ensembles of extremely randomized trees and some generic applications. Proceedings of Robust Methods for Power System State Estimation and Load Forecasting, Paris, France, pp. 1-10.

[31] Liu, Q., Liu, C. (2017). A novel locally linear KNN method with applications to visual recognition. IEEE Transactions on Neural Networks and Learning Systems, 28(9): 2010-2021. https://doi.org/10.1109/tnnls.2016.2572204

[32] Lederer, J. (2021). Linear Regression. In: Fundamentals of High-Dimensional Statistics. Springer, Cham, 37-79. https://doi.org/10.1007/978-3-030-73792-4_2

[33] Montgomery, D., Peck, E.A., Vining, G.G. (2015). Introduction to Linear Regression Analysis. John Wiley & Sons, New York, USA, 14-42.

[34] Khalaf, G. (2022). Improving the ordinary least squares estimator by ridge regression. Open Access Library Journal, 9(5): 1-8. https://doi.org/10.4236/oalib.1108738

[35] Haversine Formula. https://www.movable-type.co.uk/scripts/latlong.html, accessed on Aug. 25, 2022.

[36] Piepho, H.P. (2019). Ridge regression and extensions for genome wide selection in maize. Crop Science, 49(4): 1165-1176. https://doi.org/10.2135/cropsci2008.10.0595

[37] Gebken, B., Bieker, K., Peitz, S. (2022). On the structure of regularization paths for piecewise differentiable regularization terms. Journal of Global Optimization, 85: 709-741. https://doi.org/10.1007/s10898-022-01223-2

[38] Jomthanachai, S., Wong, W.P., Khaw, K.W. (2022). An application of machine learning regression to feature selection: a study of logistics performance and economic attribute. Neural Computing and Application, 34(8): 15781-15805. http://doi.org/10.1007/s00521-022-07266-6

[39] Wang, W., Liang, J., Liu, R., Song, Y., Zhang, M. (2022). A robust variable selection method for sparse online regression via the elastic net penalty. Mathematics, 10(16): 2985. https://doi.org/10.3390/math10162985

[40] Sahebalam, H., Gholizadeh, M., Hafezian, H., Ebrahimi, F. (2022). Evaluation of bagging approach versus GBLUP and Bayesian LASSO in genomic prediction. Journal of Genetics, 101(1): 19. http://doi.org/10.1007/s12041-022-01358-x

[41] Bearing Formula. https://www.movable-type.co.uk/scripts/latlong.html, accessed on Aug. 25, 2021.

[42] Gonzalo-Martin, C., Lillo-Saavedra, M., Garcia-Pedrero, A., Lagos, O., Menasalvas, E. (2017). Daily evapotranspiration mapping using regression random forest models. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 10(12): 5359-5368. https://doi.org/10.1109/JSTARS.2017.2733958

[43] Zhao, L., Lee, S., Jeong, S.P. (2021). Decision tree application to classification problems with boosting algorithm. Electronics, 10(16): 1903. https://doi.org/10.3390/electronics10161903

[44] Rahmatian, M., Chen, Y.C., Palizban, A., Moshref, A., Dunford, W.G. (2017). Transient stability assessment via decision trees and multivariate adaptive regression splines. Electric Power Systems Research, 142: 320-328. https://doi.org/10.1016/j.epsr.2016.09.030

[45] Ilyas, Q.M., Mehmood, A., Ahmad, A., Ahmad, M.A. (2022). Systematic study on a customer's next-items recommendation techniques. Sustainability, 14(12): 7175. http://doi.org/10.3390/su14127175

[46] Koren, Y. (2009). The Bellkor solution to the Netflix grand prize. Netflix Prize Documentation, 81(2009): 1-10.

[47] Ma, H., Yang, X., Mao, J., Zheng, H. (2018). The energy efficiency prediction method based on gradient boosting regression tree. In 2018 2nd IEEE Conference on Energy Internet and Energy System Integration (EI2), Beijing, China, pp. 1-9. https://doi.org/10.1109/EI2.2018.8581904

[48] Wu, F., Fu, K., Wang, Y., Xiao, Z. (2017). A spatial-temporal-semantic neural network algorithm for location prediction on moving objects. Algorithms, 10(2): 37-42. https://doi.org/10.3390/a10020037

[49] Novatel GNSS. https://www.novatel.com/products/gnss-receivers/, accessed on Aug. 25, 2021.

[50] Ublox GNSS. https://www.u-blox.com/sites/default/files/products/documents/GNSS-product_Overview_%28UBX-14000426%29.pdf, accessed on Aug. 25, 2021.

[51] Shoab, M., Jain, K., Anulhaq, M., Shashi, M. (2013). Development and implementation of NMEA interpreter for real-time GPS data logging. In 2013 3rd IEEE International Advance Computing Conference (IACC), Ghaziabad, India, pp. 143-146. https://doi.org/10.1109/IAdCC.2013.6514210

[52] Li, S., Pischinger, S., He, C., Liang, L., Stapelbroek, M. (2018). A comparative study of model-based capacity estimation algorithms in dual estimation frameworks for lithium-ion batteries under an accelerated aging test. Applied Energy, 212: 1522-1536. https://doi.org/10.1016/j.apenergy.2018.01.008

[53] Tang, X., Zou, C., Yao, K., Chen, G., Liu, B., He, Z., Gao, F. (2018). A fast estimation algorithm for lithium-ion battery state of health. Journal of Power Sources, 396: 453-458.

[54] Abidin, D.Z., Nurmaini, S., Erwin, E., Rasywir, E., Pratama, Y. (2021). Indoor positioning system in learning approach experiments. Journal of Electrical and Computer Engineering, 2021(8). https://doi.org/10.1155/2021/6592562

[55] Ponnmoli, K.M., Selvamuthukumaran, S. (2017). Analysis of face recognition using Manhattan distance algorithm with image segmentation. International Journal of Computer Science and Mobile Computing, 3(7): 18-27.

[56] Ahmed, M., Seraj, R., Islam, S.M.S. (2020). The k-means algorithm: A comprehensive survey and performance evaluation. Electronics, 9(8): 1295. https://doi.org/10.3390/electronics9081295

[57] Ikotun, A.M., Almutari, M.S., Ezugwu, A.E. (2021). K-means-based nature-inspired metaheuristic algorithms for automatic data clustering problems: Recent advances and future directions. Applied Sciences, 11(12): 11246. https://doi.org/10.3390/app112311246

[58] Ersan, Z.G., Zontul, M., Yelmen, I. (2020). Map matching with Kalman filter and location estimation. Cumhuriyet Science Journal, 41(1): 43-48. http://doi.org/10.17776/csj.634940