POLITECNICO DI TORINO
Repository ISTITUZIONALE

Detecting Phase Scintillation at High Latitudes Using Ionospheric Scintillation Monitoring Records and Machine Learning Techniques

Original
Detecting Phase Scintillation at High Latitudes Using Ionospheric Scintillation Monitoring Records and Machine Learning Techniques / Imam, R., Savas, C., Dovis, F.. - ELETTRONICO. - (2021), pp. 37-42. (2021 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE) Cleveland, OH, USA 12-14 Oct. 2021) [10.1109/WiSEE50203.2021.9613840].

Availability:
This version is available at: 11583/2951313 since: 2022-02-02T10:47:24Z

Publisher:
IEEE

Published
DOI:10.1109/WiSEE50203.2021.9613840

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

12 September 2026

Detecting Phase Scintillation at High Latitudes Using Ionospheric Scintillation Monitoring Records and Machine Learning Techniques

Rayan Imam
Dept. of Elec. and Telecom.
Politecnico di Torino
Turin, Italy
rayan.imam@polito.it

Caner Savas
Dept. of Elec. and Telecom.
Politecnico di Torino
Turin, Italy
caner.savas@polito.it

Fabio Dovis
Dept. of Elec. and Telecom.
Politecnico di Torino
Turin, Italy
fabio.dovis@polito.it

***Abstract*—In this paper, we present a bagged tree model able to detect phase scintillation at high latitudes with 95% accuracy, 5% scintillation miss-detection and 5% scintillation false alarm. The input to the model is a series of 3 minutes of the Total Electron Content (TEC), 3 minutes of the change in TEC (dTEC), and the satellite elevation. These values are extracted from Ionospheric Scintillation Monitoring Records (ISMR) logged by Ionospheric Scintillation Monitoring (ISM) receivers. We compare the performance of this model to Support Vector Machine (SVM) models, k-Nearest Neighbors (k-NN) models, and also to other decision tree models. Furthermore, we assess the ability of the TEC and dTEC features to detect scintillation independently of the scintillation indexes. For this, we compare the above decision trees, kNN and SVM models to the same models but trained using scintillation indexes as additional inputs. Moreover, we show the results of testing the proposed model using a novel data set. Finally, we compare the accuracy of the machine learning model to the performance of a detector based on the phase scintillation index σ_ϕ threshold.**

***Index Terms*—global navigation satellite system, Supervised learning, Geophysical signal processing, bagged decision trees, support vector machines**

I. Introduction

Phase scintillations are rapid and random fluctuations in the phase of radio wave signals as they pass through irregularities in the electron density of the ionosphere. These irregularities occur mainly near the earth magnetic equator and poles causing disturbances to trans-ionospheric signals at these regions. High latitude phase scintillations are induced by ionospheric irregularities often associated to space weather events like geomagnetic storms.

Global Navigation Satellite Systems (GNSS) services are vulnerable to scintillation because scintillation leads to degraded availability, reliability and accuracy of the GNSS service. On the other hand, because GNSS signals are susceptible to scintillations, they have been utilized as signals of opportunity to monitor the state of the ionosphere. Leveraging the global coverage of multi-frequency GNSS signals, ionospheric scintillation monitoring (ISM) receivers have been continuously recording ionospheric measurements for the last couple of decades. This resulted in a rich repository of Ionospheric Scintillation Monitoring Records (ISMR), which are logged at 1 minute rate, as well as other records with higher logging rate (for example raw correlator outputs are typically logged at 50 Hz or 100 Hz and raw IF (Intermediate Frequency) GNSS signals are logged at several MHz).

For such a high volume of data, automatic detection of scintillation is necessary, and in fact it has always been implemented by ISM receivers and ionospheric studies' researchers. However, phase scintillation detection, and GNSS scintillation in general, has not been a trivial task because of the scintillation-like anomalies that affect the scintillation indexes. Satellite and receiver clock anomalies, and multipath are the main sources for false scintillation alarms in GNSS measurements [1].

In recent years, Machine Learning (ML) models to detect phase scintillation have been proposed in the literature. For example, in [2], the high-rate raw correlator measurements from ISM receivers are utilized to train machine learning models able to detect phase scintillation and the performance of SVM (Support Vector Machines)-based implementations for phase and amplitude scintillation detection is evaluated. In [3], amplitude and phase scintillation indexes provided in ISMR records are utilized to carry out the detection task. In this paper we utilize the other measurements in ISMR files to detect phase scintillation. We investigate the feasibility of detecting phase scintillation relying on Total Electron Content (TEC) and the change in TEC over time (dTEC).

The motivation behind using ISMR records is that they are available with almost continuous monitoring for decades. That makes them a rich resource of scintillation monitoring data. In this paper we focus on exploiting TEC and dTEC measurements calculated from dual frequency pseudorange measurements and carrier phase measurements, respectively. Furthermore, besides SVM [3] and decision tree [4] [5] learning methods, the usage of k-NN (k-Nearest Neighbors) algorithm for phase scintillation detection is investigated through a comparative performance analysis. Because, beforehand it's quite hard to choose directly the correct method in most cases

Identify applicable funding agency here. If none, delete this.

in high dimensional spaces on the selection of SVM, k-NN, and decision trees for the classification problems. Most of the time, a validation data-set is used to not only optimize hyperparameters of the algorithms but also to choose between algorithms, as is investigated in this paper.

The rest of the paper is organized as follows. In Section II, we review the machine learning algorithms investigated in this paper. In Section III we present the trained machine learning models and show the results of testing these models. We conclude the paper in Section IV summarizing the results.

II. Overviews of The Applied ML Approaches to Scintillation Detection

Support Vector Machines (SVM) is a model-based algorithm creating a model of which the parameters are learned from the training data. However, k-Nearest Neighbors (k-NN) is an instance-based algorithm that uses the whole dataset as the model [6]. Both methods can be used for classification and regression, and in classification cases, given a feature vector, they only output the class. Furthermore, decision tree learning is another type of classification and regression algorithm that also provides a score beside the class output, as being different from SVM and k-NN. It shows the confidence level of the algorithm on the prediction made for a certain class [6]. Overviews of the algorithms considered for scintillation detection (i.e. classification) in this paper are given in the following.

A. SVM Algorithm

SVM algorithm that is one of the most influential supervised learning approach associated with kernel trick aims to classify the samples using a separating hyperplane and it output a class identity [7]. Fig. 1 depicts an example of SVM for two-class (e.g. square and circle samples) classification.

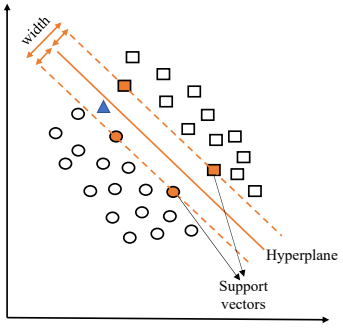

Fig. 1. An overview sketch of SVM algorithm linear classifier.

The predictions are made using the function [7]

$$f(x) = b + \sum_{i=1}^{N_s} w_i \, \kappa(x, x^i) \tag{1}$$

where $f(x)$ describes the approximate relationship between input x and corresponding target output values y. x^i is a sample of training input data-set given $\left\{x^1, x^2, \cdots, x^{N_s}\right\}$ where $x^i \in \mathbb{R}^n$. b is the parameter of the optimum hyperplane shown in Fig. 1 and w_i is the vector coefficients, which are to be optimized. κ is the kernel function. In some cases in which the samples are linearly separable, the linear kernel function can be used:

$$\kappa(x_i, x_j) = x_i^T x_j \tag{2}$$

where the kernel function κ provides a mapping from the instance space to a feature space associated the kernel. Hence, it enables to find another hyperplane in the kernel space and to achieve nonlinear separation in the feature space [8]. The most commonly used kernel is the Gaussian kernel also known as Radial Basis Function (RBF):

$$\begin{aligned} \kappa(x_i, x_j) &= \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \\ &= \exp\left(-\gamma \|x_i - x_j\|^2\right) \end{aligned} \tag{3}$$

where σ is the width of the kernel and it has to be properly selected. If it gets closer to zero, it might cause over-fitting. However, a bigger value of σ might lead to under-fitting and ends up with classifying all the instances into one class [8]. Selection of kernel scale parameter γ in the RBF has the similar issues as well. Another well-known kernel function is the polynomial the polynomial:

$$\kappa(x_i, x_j) = \left(1 + x_i^T x_j\right)^p \tag{4}$$

where addition of 1 provides in-homogeneity

In Section III, a comparative performance analysis of linear kernel, Gaussian kernel having different kernel scale parameters, and polynomial kernel with second and third order functions through experimental test results is provided.

B. k-NN Algorithm

k-NN is a type of non-probabilities supervised algorithm and it is generally used for classification or regression. In classification, k-NN algorithm looks at the close neighborhood of the input example in the feature space and labels it that as seen in this close neighborhood [6]. In other words, a test sample ($x^{(i)}$) is classified considering majority class of its neighbors:

$$y = \arg\max_{t^{(i)}} \sum_{r=1}^{k} d(x^{(i)}, t^{(r)}) \tag{5}$$

where $t^{(i)}$ is a class label and d is the distance metric. The algorithm is generalized through a distance metric to measure the distance or similarity between training samples and test examples. Fig. 2 shows an example of k-NN classification cases for $k = 1$ and $k = 3$, where k is the number of training examples closest to the considered input sample.

As it is observed in Fig. 2, if $k > 1$, there are multiple training samples describing an example input test sample that is shown as a blue triangle mark. When $k = 1$, it creates a locally constant surface computed cell-by-cell in which the

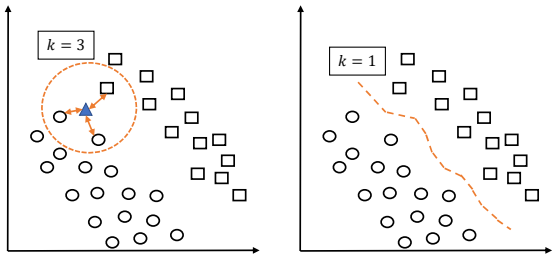

Fig. 2. k-NN classification example.

feature space is divided. There are various distance metrics and one of them used for real-valued vector spaces is Minkowski distance [9]:

$$d(x_i, x_j) = \left(\sum_{i=1}^{l} |x_i - x_j|^r \right)^{1/r} \tag{6}$$

where l is the number of dimensions. x_i and x_j are the data points. When $p = 1$ it is Manhattan distance and if p is set to 2, Euclidean distance that is most popular among distance metrics is got. It represents the root of the sum of the square of differences in vectors [9]. Euclidean distance can be weighted

$$d(x_i, x_j) = \sqrt{\sum_{i=1}^{m} w_i \left(x_i - x_j\right)^2} \tag{7}$$

where w_i is the weight that influences the distance of instance $x_i = [x_{i1}, x_{i2}, \cdots, x_{im}]$ to the nearest neighbor instance $x_j = [x_{j1}, x_{j2}, \cdots, x_{jm}]$. w_i represents the feature weighting that consists of m weight coefficient for m features. Furthermore, a distance weighted k-NN can also be applied directly in (5) before distance function $d(x^{(i)}, t^{(r)})$. One of the widely applied weighting is the inverse squared distance.

Cosine distance that is also called angular distance is a type of similarity measure [9]:

$$d(x_i, x_j) = \frac{\vec{x_i} \cdot \vec{x_j}}{\|\vec{x_i}\| \, \|\vec{x_j}\|} \tag{8}$$

where $\cdot$ represents the dot product between two vectors and it is normalized by their magnitude.

The choice of the distance metric and the value of k should be made carefully. In order have powerful k-NN classification, a proper value for k should be selected. If it is set too small, k-NN becomes sensitive to class noise. However, selecting k too large leads to include many neighbor points and hence increases the bias. A performance comparison of selected different k values and distance metrics in terms of scintillation detection accuracy through carried out experimental tests is discussed in Section III.

C. Decision Tree Learning

Decision tree learning is based on tree structures, defined by recursively partitioning the input space, as depicted in Fig. 3. Decision tree is an acyclic graph in which each branching node a decision is made by examining a specific feature vector and depending on the decision the right or left branch is followed [6]. In other words, the learning takes place along the branches and nodes by means of applied functions for the decision criteria in each node [4].

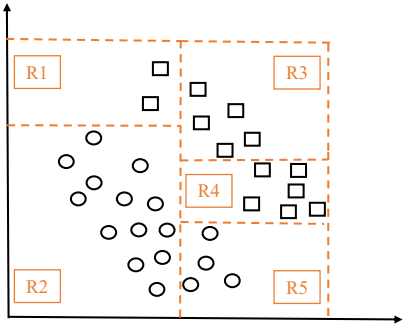

Fig. 3. An example of feature space partitioning in decision tree.

A classifier can be represented by $C(S, x)$ where x is the input point and S is the training data having a set of labeled data:

$$S \stackrel{\text{def}}{=} \{(x_i, y_i)\}_{i=1}^{N} \tag{9}$$

where it also denotes the start node that contains all examples [6].

Bagging trees utilize an ensemble technique that creates a classifier from training a number of tree classifiers. Bagging trees classify by majority vote

$$C(x) = \text{Majority Vote} \left\{ C\left(S^{(b)}, x\right) \right\}_{b=1}^{B} \tag{10}$$

where B is the number of decision trees in the ensemble.

Boosting is another ensemble technique that classifies by weighted majority vote

$$C(x) = \text{sign} \left[\sum_{m=1}^{M} w_m \, C_m(x) \right] \tag{11}$$

where M is the number of decision trees in the ensemble.

The tree structure should be pruned to an optimal size through evaluations of cross-validation results.

III. Implementation and Test Results

ISMR data collected in Antarctica was utilized to train and test the machine learning models. In this section we describe the data preparation for the machine learning task. Then we present the machine learning models obtained in this paper. Finally we show and discuss the testing results of testing these models.

A. Data Preparation

The data utilized to train, validate and test the models were collected at the South African Antarctic research base (SANAE IV, 71.67 S, 2.84 W) using Septentrio PolaRx5S Receiver between 23-29 August 2018. These days were selected

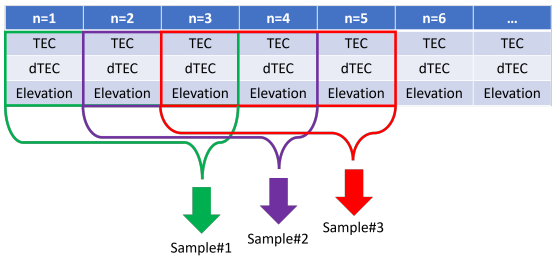

Fig. 4. Data preparation.

because on 26-27 August 2018 phase scintillation occurred at high latitudes due to a G3 geomagnetic storm [10].

The data was manually labeled using visual inspection after consulting data from consecutive days (to check for multipath), data from all visible satellites (to check for receiver clock errors), and data from different frequency bands (to check for satellite clock errors). To read more about checking for multipath and oscillator anomalies refer to [5] and [11] respectively. The data is labeled as scintillated if the phase measurements were randomly fluctuating, but no multipath or clock anomaly was observed during the anomaly check mentioned above. Otherwise, the data was labeled as non-scintillated. Approximately $16,000$ labeled samples were extracted, of which $50\,\%$ were scintillated. The samples were randomly split into training and testing sets, with 70% of the data in the training set. Again, 50% of the training set was scintillation. Validation using 5-fold cross-validation was also implemented during the training phase.

Two sets of attributes were considered for training the machine learning models:

- F1: contains a 3-minutes series of TEC, dTEC and elevation values. A sliding window step of 1 minute was implemented. In other words, the label is given at a rate of 1 minute, and we evaluate 3 minutes of data to give the label. TEC and dTEC are logged in ISMR files every 15s.
- F2: contains all the measurements related to L1 CA signal in the ISMR file, also grouped in blocks of 3-minutes overlapping measurements, with sliding window step of 1 minute.

Fig. 4 depicts how the 3-minute overlapping window was prepared. In this figure, we showed only 3 features for demonstration purpose, but the concept is general.

In [12], the analysis carried out through collected data shows that the mean duration of the phase scintillation events in the polar region is 5.6 minutes. So as not to miss shorter scintillation events too, time-window is adjusted to 3-min through an experimental analysis. Furthermore, the tests carried out applying an extended time window (e.g. 5-min) has not provided a significant improvement in the accuracy.

B. The Machine Learning Models

Table I summarizes the applied ML models having different settings. The first column shows the general machine learning models name. The second column shows the name of the variation of the model. By model variation we mean different hyperparameter settings of the model. The third column describes the model variation indicating the hyperparameters values.

TABLE I
THE MACHINE LEARNING MODELS CONSIDERED

Model	Variation	Description
Decision Tree Learning	Fine	Max number of splits = 100
	Medium	Max number of splits = 20
	Coarse	Max number of splits = 4
	Bagged	Ensemble method: Bag, Number of learners = 30 Max number of splits = No limit
	Boosted	Ensemble method: AdaBoost Number of learners = 30 Max number of splits = 20
SVM	Linear	Kernel function: Linear
	Quadratic	Kernel function: Quadratic
	Cubic	Kernel function. Cubic
	Fine	Kernel function: Gaussian Kernel Scale = $\sqrt{(n)}/4$ where n is the number of predictors
	Medium	Kernel function: Gaussian Kernel Scale = $\sqrt{(n)}$
	Coarse	Kernel function: Gaussian Kernel Scale = $4\sqrt{(n)}$
k-NN	Fine	Distance metric: Euclidean Number of neighbors = 1
	Medium	Distance metric: Euclidean Number of neighbors = 10
	Coarse	Distance Metric: Euclidean Number of neighbors = 100
	Cosine	Distance metric: Cosine Number of neighbors = 10
	Cubic	Distance metric: Minkowski Number of neighbors = 10
	Weighted	Distance metric: Euclidean Number of neighbors = 10 Distance weight : Square inverse

C. Experimental Test Results

Figure 5 reports the results of training and testing the models using feature set F1, while Fig. 6 reports the results of training and testing the models using Feature set F2. The x-axis reports the models. The solid and dashed blue lines report the training and testing accuracy, respectively. The solid and dashed red lines report the miss-detection and false alarm rate respectively for the scintillation class. These metrics are defined as follows:

- Accuracy is the ratio of the number of correctly classified samples to the total number of samples.
- Scintillation Miss detection rate is the ratio of the number of scintillation samples wrongly classified to the total number of scintillation samples.

- False scintillation alarm is the ratio of the number of samples wrongly classified as scintillation to the total number of samples classified as scintillation.

Focusing on the bagged tree models (BaggedT) in Fig. 5 and Fig. 6, the results show that it is possible to obtain a model with 95% accuracy, 5% false alarm, and 5% scintillation miss detection, relying on TEC, dTEC and satellite elevation measurements alone (i.e. F1). Moreover, using F2 which includes all ISMR measurements did not give superior results to using F1 only. This indicates that TEC and dTEC are utilizable to detect phase scintillation without the need for phase scintillation indexes. Furthermore, many of the machine learning models achieved comparable good results in terms of accuracy, however the bagged tree demonstrated the highest accuracy.

The k-NN models in general reported lower accuracy and higher scintillation miss-detection than the trees. The decrease observed in the accuracy of coarse k-NN is expected since the number of neighbors might be accepted as the limit considering the rule of thumb (i.e. $k < \sqrt(m)$, where m is the number of training examples). At the same time, the cosine distance function, which is used when the magnitude between vectors does not matter but the orientation, could track better the trend of the scintillation indices (i.e., consecutive decrease/increase).

Finally, the SVM models, except the cubic SVM, reported a performance comparable to the trees. A drop in the accuracy of the cubic SVM is expected at an acceptable range observed in F1 and F2, considering the possibility that higher-degree polynomial might lead to over-fitting in the training test set. However, the same is not valid in cubic k-NN, because the classification output is computed through the majority class in that case. Furthermore, Manhattan distance has not been applied since it would work as thresholding.

D. Comparison with Standard Method

We compare the performance of the bagged tree model trained with F1 to the performance of a standard method in the literature; the threshold on the phase scintillation metric σ_{phi}. For the Septentrio PolaRx5S Receiver installed at SANAE IV station, a threshold of $\sigma_\phi = 0.15\ rad$ is acceptable to detect moderate scintillation. To detect weak scintillation, lower threshold value is needed. For this reason, we tested different threshold values: 0.05, 0.1 and $0.15\ rad$ as shown in Table II. In the table we report the scintillation miss-detection and false alarm beside the overall accuracy. We can see that with the threshold technique, the accuracy is below 80% while with the bagged tree model it reaches $\sim 95\%$. The improvement comes mainly from the significantly low miss-detection rate of the machine learning model (5.2%) compared to 27.8% for the best case in threshold method. The false alarm is slightly higher than a 0.1 rad threshold, but it is acceptable when looking at the overall accuracy.

TABLE II
COMPARING THE PERFORMANCE OF THE BAGGED TREE MODEL TO STANDARD THRESHOLD METHOD

Method	Miss detection [%]	False alarm [%]	Accuracy [%]
Threshold=0.05	27.8	16.5	79.0
Threshold=0.10	57.2	3.2	70.7
Threshold=0.15	71.4	1.3	64.1
Bagged Tree Model	5.2	5.4	94.7

IV. CONCLUSIONS

In this paper, we compare the performance of SVM, k-NN and Tree models trained to detect high latitude phase scintillation. We propose relying on series of 3-minutes TEC and dTEC measurements extracted directly from ISMR files to train the models. We compare these models to models trained with 3-minute samples that contain all L1CA related measurements in the ISMR records, in addition to the TEC and dTEC values. Finally, we show examples of testing the models with a novel data set.

We show that bagged trees relying on 3-minutes of TEC and dTEC measurements are able to detect phase scintillation with 95% accuracy, 5% scintillation miss detection and 5% scintillation false alarm.

REFERENCES

[1] J. Vilà-Valls, N. Linty, P. Closas, F. Dovis, and J. T. Curran, "Survey on signal processing for GNSS under ionospheric scintillation: Detection, monitoring, and mitigation," *NAVIGATION*, vol. 67, no. 3, pp. 511–536, 2020. [Online]. Available: https://doi.org/10.1002/navi.379

[2] Y. Jiao, J. J. Hall, and Y. T. Morton, "Performance evaluation of an automatic GPS ionospheric phase scintillation detector using a machine-learning algorithm," *NAVIGATION*, vol. 64, no. 3, pp. 391–402, 2017.

[3] C. Savas and F. Dovis, "The impact of different kernel functions on the performance of scintillation detection based on support vector machines," *Sensors*, vol. 19, no. 23, 2019.

[4] N. Linty, A. Farasin, A. Favenza, and F. Dovis, "Detection of GNSS ionospheric scintillations based on machine learning decision tree," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 1, pp. 303–317, 2019.

[5] R. Imam and F. Dovis, "Distinguishing ionospheric scintillation from multipath in gnss signals using bagged decision trees algorithm," in *2020 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, 2020, pp. 83–88.

[6] A. Burkov, *The Hundred-page Machine Learning Book*. Andriy Burkov, 2019. [Online]. Available: https://books.google.it/books?id=ZF3KwQEACAAJ

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[8] Z. Liu, M. J. Zuo, X. Zhao, and H. Xu, "An analytical approach to fast parameter selection of gaussian rbf kernel for support vector machine." *J. Inf. Sci. Eng.*, vol. 31, no. 2, pp. 691–710, 2015.

[9] H. A. Abu Alfeilat, A. B. Hassanat, O. Lasassmeh, A. S. Tarawneh, M. B. Alhasanat, H. S. Eyal Salman, and V. S. Prasath, "Effects of distance measure choice on k-nearest neighbor classifier performance: a review," *Big data*, vol. 7, no. 4, pp. 221–248, 2019.

[10] F. Palma and et al., "The august 2018 geomagnetic storm observed by the high-energy particle detector on board the CSES-01 satellite," *Applied Sciences*, vol. 11, no. 12, 2021.

[11] Y. Liu and Y. J. Morton, "Automatic detection of ionospheric scintillation-like gnss satellite oscillator anomaly using a machine-learning algorithm," *NAVIGATION*, vol. 67, no. 3, pp. 651–662, 2020.

[12] Y. Jiao, Y. Morton, and S. Taylor, "Comparative studies of high-latitude and equatorial ionospheric scintillation characteristics of gps signals," in *2014 IEEE/ION Position, Location and Navigation Symposium - PLANS 2014*, 2014, pp. 37–42.

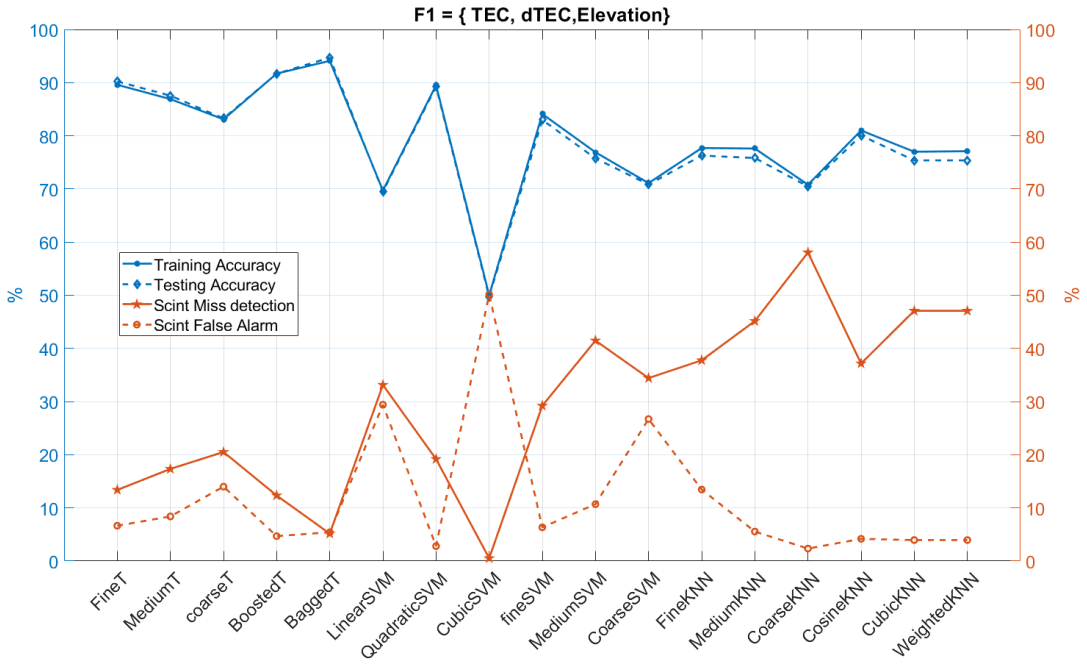

Fig. 5. Results of training and testing the models using feature set F1.

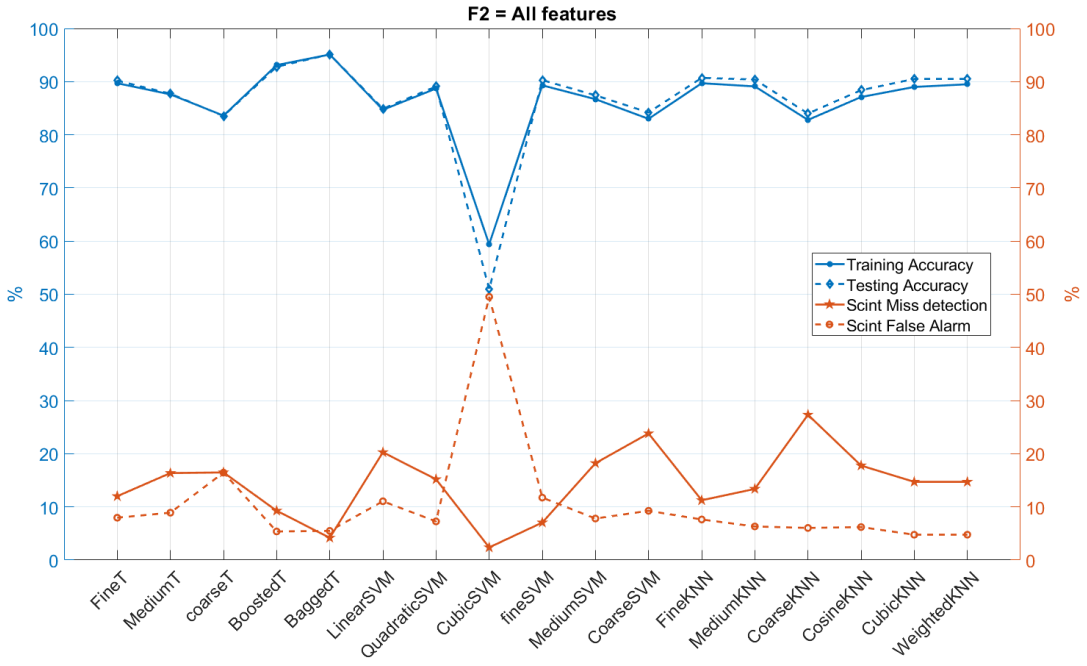

Fig. 6. Results of training and testing the models using feature set F2.