A Dst-based space weather conditions machine learning classification model for GNSS PNT performance analysis

Deni Klen

Lucija Žužić

Teodor B. Iliev

Renato Filjar

Abstract

Ambient conditions classification enables systematic mitigation of adversarial effects on the Global Navigation Satellite System (GNSS) Positioning, Navigation, and Timing (PNT) performance. This research contributes to the problem by proposing a classification model of space weather events for sub-equatorial regions. The proposed model uses machine learning-based classification applied to the experimental observations of geomagnetic field components, observed Total Electron Content, and Disturbance Storm-Time (Dst) index. The Least Squares Support Vector Machine (SVM) with a Polynomial Kernel, the C5.0 Decision Tree (DT), the Naive Bayes, the Neural Network, and the Partial Least Squares methods are applied to develop the candidate classification model to classify a set of observations of the geomagnetic field in TEC into one of the scenarios of space weather conditions. Candidate models’ performance is assessed for accuracy using a confusion matrix, and model development time to yield the Naive Bayes as the best-performing model. The proposed Dst-based classification model may serve as an indicator of a geomagnetic/ionospheric storm in progress, thus alerting GNSS users of a potential degradation in GNSS PNT performance, and setting up a framework for the development of a tailored GNSS ionospheric correction model for specific classes of the space weather conditions.

Global Navigation Satellite System (GNSS) Positioning ,Navigation, and Timing (PNT) ,space weather conditions ,machine learning ,Total Electron Content (TEC) ,Disturbance Storm-Time (Dst) index

# Introduction

Cause-effect relations between space weather & geomagnetic conditions drive the ionospheric conditions, which cause the GNSS ionospheric delay, and, consequently, degradations of the GNSS Positioning, Navigation, and Timing (PNT) performance (service quality) – Space weather – GNSS performance Coupling Model (Filić, Filjar, 2018) . A need for classification of geomagnetic/ionospheric conditions, regarding GNSS PNT effect intensity.

# Method and Data

## Method

Data pre-processing is recommended to increase classification accuracy . Minimum-maximum, normalization by decimal scaling, and Z-score are often used for standardization. Subtracting the mean and dividing by the variance for each feature is commonly used for Support Vector Machines (SVMs) . All Neural Network models were applied based on research by Kuhn for the *R* *caret* package .

### Neural Networks

The neurons of human or animal brains provide the basis for a Neural Network (NN) or Artificial Neural Network (ANN) with connected units or nodes called artificial neurons in machine learning . Like biological synapses, edges form links between artificial neurons that receive, process, and transmit data. Real numbers form the input, and a non-linear activation function of the sum of the inputs shapes the output. Weights are modified during training and represent signal strength at the link output. Multiple intermediate hidden layers are typically introduced to transform the data from the first input layer before it reaches the final output layer.

Shallow Neural Networks are simple and easy to train quickly with fewer parameters, and computational resources. They typically contain only a few hidden layers for processing between the input layer that receives the data, and the final layer that produces the output . A network with at least two hidden layers is considered a deep Neural Network. Artificial Neural Networks can extrapolate a solution by training on a dataset with no obvious connections, too complicated for human analysts. Artificial intelligence is used for forecasting, adaptive control, and other tasks.

Neural Networks are typically trained through empirical risk minimization. This method optimizes the network’s parameters to minimize the difference or empirical risk, between the predicted output and the actual target values in a given dataset . Gradient-based methods such as backpropagation are usually used to estimate network parameters . During the training phase, ANNs learn from labeled training data by iteratively updating their parameters to minimize a defined loss function . This method allows the network to generalize to unseen data.

Using Artificial Neural Networks requires an understanding of their characteristics. The model suitable for a specific task varies depending on how the data is stored and other practical considerations. Many model parameters may need to be adjusted and differ for each model. The type and number of layers, as well as their connections, need to be defined. Each layer has a size, and each connection has a type, including full, pooling, and others. A more advanced model might be more successful but will take longer to execute and take up more storage space. The hyperparameters for the learning algorithm may also be modified to suit the problem during an extensive tuning process experimenting on unseen training data, and various learning algorithms are available. If training is done according to recommended guidelines the model will be robust and perform adequately on additional tests.

The network design process can be simplified using previously defined machine learning techniques like neural architecture search (NAS). These networks have performed well even when pitted against architectures selected by humans. The proposed model is tested and the responses are used to train the NAS network . AutoML and AutoKeras provide extensive frameworks for this process. The *scikit-learn* library features various classification, regression, and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means, and Density-Based Spatial Clustering (DBSCAN). These network models are ready to be used and trained once the data and hyperparameters are provided. The hyperparameters must be predefined, are not trained, and include how many neurons are in each layer, learning rate, step, stride, depth, receptive field, and padding for CNNs . TensorFlow and Keras are used to build a custom network in Python, and the user can define layers, models, or metrics. The data, number of hidden layer units, learning rate, and number of iterations are the required parameters for the Python training function in Table [1](#tab:train).

An overview of the training function, which uses the training dataset, number of hidden layer units, learning rate, and number of iterations as parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | **def** train(X, y, n\_hidden, learning\_rate, n\_iter): | | | |
| 2 |  | m, n\_input = X.shape | | |
| 3 |  | | | |
| 4 |  | *# 1. random initialize weights and biases* | | |
| 5 |  | w1 = np.random.randn(n\_input, n\_hidden) | | |
| 6 |  | b1 = np.zeros((1, n\_hidden)) | | |
| 7 |  | w2 = np.random.randn(n\_input, 1) | | |
| 8 |  | b2 = np.zeros((1, 1)) | | |
| 9 |  | | | |
| 10 |  | *# 2. in each iteration, feed all layers with the latest weights and biases* | | |
| 11 |  | **for** i **in** range(n\_iter + 1): | | |
| 12 |  |  | z2 = np.dot(X, w1) + b1 | |
| 13 |  |  | a2 = sigmoid(z2) | |
| 14 |  |  | z3 = np.dot(a2, w2) + b2 | |
| 15 |  |  | a3 = z3 | |
| 16 |  |  | dz3 = a3 - y | |
| 17 |  |  | dw2 = np.dot(a2.T, dz3) | |
| 18 |  |  | db2 = np.sum(dz3, axis=0, keepdims=**True**) | |
| 19 |  |  | dz2 = np.dot(dz3, w2.T) \* sigmoid\_derivative(z2) | |
| 20 |  |  | dw1 = np.dot(X.T, dz2) | |
| 21 |  |  | db1 = np.sum(dz2, axis=0) | |
| 22 |  | | | |
| 23 |  |  | *# 3. update weights and biases with gradients* | |
| 24 |  |  | w1 -= learning\_rate \* dw1 / m | |
| 25 |  |  | w2 -= learning\_rate \* dw2 / m | |
| 26 |  |  | b1 -= learning\_rate \* db1 / m | |
| 27 |  |  | b2 -= learning\_rate \* db2 / m | |
| 28 |  | | | |
| 29 |  |  | **if** i % 1000 == 0: | |
| 30 |  |  |  | print("Epoch", i, "loss: ", np.mean(np.square(dz3))) |
| 31 |  | | | |
| 32 |  | model = {"w1": w1, "b1": b1, "w2": w2, "b2": b2} | | |
| 33 |  | **return** model | | |

### Support Vector Machine

In machine learning, Support Vector Machines (SVMs) or Support Vector Networks (SVNs) are supervised maximum margin models with associated learning algorithms that analyze data for classification and regression. The Vapnik–Chervonenkis theory developed in 1964 provided the basis for the original SVM algorithm developed at AT&T Bell Laboratories. In addition to linear classification, SVMs are also effective for non-linear classification using the hyperplane kernel trick with maximum margin developed in 1992 by Bernhard Boser, Isabelle Guyon, and Vladimir Vapnik , implicitly mapping their inputs to high-dimensional feature spaces.

SVMs can also be used for regression tasks where the objective becomes -sensitive. A "soft margin" to separate data that is not linearly separable was proposed by Corinna Cortes and Vapnik in 1993 and published in 1995 . Support Vector Clustering (SVC), developed by Hava Siegelmann and Vladimir Vapnik, applies support vector statistics to unlabeled data in unsupervised learning to find new natural groupings. The popularity of SVMs is due to their amenability to theoretical analysis, flexibility, and application to a wide range of tasks, including structured prediction problems. It is unclear whether SVM predictions perform better than other linear models, such as logistic, and linear regression.

Data classification is a common task in machine learning. Suppose that each of the given data points belongs to one of two classes, and the goal is to decide which class the new data point will belong to. In the case of Support Vector Machines, a data point is viewed as a -dimensional vector, a list of numbers, and we want to know if such points can be separated using a -dimensional hyperplane. This is called a linear classifier. Many hyperplanes could classify the data, but a reasonable choice is the one that represents the largest separation, or margin, between the two classes. In this case, the distance of the hyperplane to the nearest data point on each side is maximal. Such a hyperplane is known as a maximum margin hyperplane, and the linear classifier is a maximum margin classifier or an optimal stability perception.

SVMs belong to the family of generalized linear classifiers and can be interpreted as an extension of perceptrons. They can also be considered a special case of Tikhonov’s regulation. A special property is that they simultaneously minimize the empirical classification error and maximize the geometric margin, and are also known as maximum margin classifiers. Meyer, Leisch, and Hornik compared SVM with other classifiers .

Potential disadvantages of SVMS include fully labeling the input data. SVMs are derived from Vapnik’s theory, which avoids probability estimation on finite data, so class membership probabilities are uncalibrated. SVM is directly applicable only for two-class tasks. Therefore, algorithms that reduce the multi-class task to several binary problems must be applied to build a multi-class SVM. The parameters of the solved model are difficult to interpret.

Data pre-processing, or standardization, is recommended to increase the classification accuracy . Minimum-maximum, normalization by decimal scaling, and Z-score are among many standardization methods. Subtracting the mean and dividing by the variance for each feature is commonly used for SVM .

More formally, a Support Vector Machine constructs a hyperplane or set of hyperplanes in a high-dimensional or infinite-dimensional space. Hyperplanes are used for classification, regression, or other tasks such as outlier detection . Intuitively, a good separation is achieved by the hyperplane with the greatest distance to the nearest point in the training data belonging to any class (the so-called functional margin), because the generalization error of the classifier decreases with increasing margin . A lower generalization error means overfitting to the training data set is less likely to occur.

While the original problem can be posed in a finite-dimensional space, the sets to be distinguished are not always linearly separable. For this reason, it was proposed to map the original finite-dimensional space into a multidimensional space, assuming that the separation will be easier in that space. To keep the computational burden reasonable, mappings using SVM schemes are designed to ensure that the products of matrix multiplication of pairs of input vectors can be easily computed in terms of variables in the source space, defining them in terms of a kernel probability density function chosen to fit the problem .

Hyperplanes in multidimensional space are points whose vector product with a vector in that space is constant. Such a set of vectors is an orthogonal, and thus minimal, set of vectors defining the hyperplane. The vectors defining the hyperplanes can be chosen as linear combinations with the parameters of the feature vector images appearing in the database. With this choice of hyperplane, the points in the feature space that are mapped in the hyperplane are defined by the expression in Equation [[eqn:1]](#eqn:1), which has a constant sum.

It should be noted that if becomes small as moves away from , each term in the sum measures the degree of closeness of the test point to the corresponding point in the database . The above sum of kernel probability density function values is used to measure the relative proximity of each test point to data points originating from one or the other set to be discriminated between. The set of points mapped to any hyperplane can be complex, allowing for more sophisticated discrimination between sets that are not convex in the original space.

SVMs have been used to solve a variety of real-world problems. SVMs are useful in text and hypertext categorization, as their application can significantly reduce the need for labeled training instances in standard inductive and transductive settings . Some methods for shallow semantic parsing are based on SVMs . Image classification can also be performed using SVM. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after only three to four rounds of relevance feedback. This is also true for image segmentation systems, including those using a modified version of SVM that uses privileged access as proposed by Vapnik .

Classification of satellite data such as Synthetic Aperture Radar (SAR) data is possible using a supervised SVM . Handwritten characters can be recognized using SVM . The SVM algorithm is widely used in biological and other sciences. They were used to classify proteins with up to compounds correctly classified. Permutation tests based on SVM weights have been proposed as a mechanism for interpreting SVM models . SVM weights were also used to interpret SVM models . The subsequent interpretation of SVM models to identify features used by the model for prediction is a relatively new area of research of particular importance in the biological sciences.

Let there be a training data set with points for which is or , and each denotes the class to which the point belongs. Each is a -dimensional real vector. We want to find the hyperplane with the maximum margin that separates the group of points for which holds from the group of points for which holds. A hyperplane with a maximum margin is defined so the distance between the hyperplane and the nearest point from any group is maximized.

Any hyperplane can be written as a set of points that satisfy the expression . The expression denotes a vector perpendicular to the hyperplane which is not necessarily normalized. This is similar to the Hesse normal form, except that is not necessarily a unit vector. The parameter in Equation [[eqn:2]](#eqn:2) determines the displacement of the hyperplane from the origin along the normal vector .

The original maximum margin hyperplane algorithm proposed by Vapnik in 1963 constructed a linear classifier. However, in 1992, Bernhard Boser, Isabelle Guyon, and Vladimir Vapnik proposed creating nonlinear classifiers using the kernel trick. The kernel trick on the maximum margin hyperplane was first proposed by Aizerman et al. . A nonlinear kernel probability density function replaces each matrix multiplication product. This allows the algorithm to fit the maximum margin hyperplane in the transformed feature space. The transformation can be non-linear, and the transformed space can have many dimensions. Although the classifier is a hyperplane in the transformed feature space, it may be nonlinear in the original input space. A multidimensional feature space increases the generalization error of Support Vector Machines, so many samples are needed to enhance performance .

Some common kernel probability density functions include the polynomial (homogeneous), which becomes linear for , , polynomial (inhomogeneous), , and the Gaussian Radial Basis Function (RBF), for which , . RBF can be parameterized as . The sigmoid function (hyperbolic tangent) has the equation for some values of , but not for all, and for . The equation links the kernel to the transformation . The value in Equation [[eqn:3]](#eqn:3) is also located in the transformed space.

The matrix product with for classification can be calculated using the kernel trick in Equation [[eqn:4]](#eqn:4).

The performance of SVM depends on the kernel probability density function, the parameters of the specified kernel probability density function, and the "soft" margin parameter . The Gaussian kernel probability density function, which has one parameter , is often used. The best combination of and is selected by searching a grid with exponentially increasing sequences of and , for example, , . Each combination of parameter choices is usually cross-validated, and the parameters with the best accuracy are selected. Recent advances in Bayesian optimization can be used to choose and . This method requires the evaluation of many fewer combinations of parameters than when searching the entire network. The final model is used to test and classify new data and is trained using the selected parameters .

### Decision Tree

Decision Trees are an approach to supervised learning in statistics, data mining, and machine learning. They represent predictive or decision-making model used to make inferences about a series of observations. Tree models in which the target variable can take on a discrete set of values are called classification trees. In these tree structures, the leaves represent the class labels and the branches represent the combinations of features that lead to those class labels. Decision Trees where the target variable can take on continuous values (usually real numbers) are called regression trees. More generally, the concept of a regression tree can be extended to any object equipped with pairwise differences such as categorical arrays .

Decision Trees are among the most popular machine learning algorithms due to their comprehensibility and simplicity . A Decision Tree can visually and explicitly represent decisions and decision-making in decision analysis. The Decision Tree describes the data in data mining, but the resulting classification tree can be the input for decision-making. Decision Tree learning is often used in data mining . The goal is to create a model that predicts the value of a target variable based on several input variables.

A Decision Tree is a simple representation for classifying examples. For this text, assume that all input features have finite discrete domains and that there is a single target feature called "classification". Each element of the classification domain is called a class. A decision or classification tree is one in which each internal node, not a leaf, is labeled with an input feature. Arcs coming from a node labeled by an input feature are labeled with each of the possible values of the target feature, or an arc leads to a child decision node on another input feature. Each leaf of the tree is labeled with a class or a probability distribution over the classes, which means that the tree has classified the data set into a particular class or probability distribution. The probability distribution is skewed toward certain subsets of classes if the Decision Tree is well constructed.

The tree is built by dividing the original set, which forms the root node, into subsets that form descendants, or successors. The segmentation is made using a set of rules based on classification features . This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is complete when the subset at a node all have the same target variable values or the division no longer adds value to the predictions. This process of Top-Down Induction of Decision Trees (TDIDT) is an example of a greedy algorithm and is by far the most common strategy for learning Decision Trees from data . C5.0, used in the *caret* package in R, has a similar approach and improves the ID3 and C4.5 algorithms.

Decision Trees can also be described as a combination of mathematical and computational techniques that help describe, categorize, and generalize a given data set in data mining. Data comes in records of the form . The dependent variable is the target variable we need to understand, classify, or generalize. Features form the vector used for that task.

### Naive Bayes

In statistics, Naive Bayes (NB) classifiers are a family of linear "probabilistic classifiers" that assume that, given a target class, the features are conditionally independent. The classifier got its name from this strong and possibly naive assumption. These classifiers are among the simplest Bayesian network models . Naive Bayes classifiers are highly scalable and require several parameters. The number of parameters depends linearly on the number of variables. Variables represent features or predictors in the learning problem. Maximum likelihood training can be performed by evaluating a closed-form expression , which requires linear time, instead of the expensive iterative approximation used for many other classifiers.

In the statistical literature, Naive Bayes models are known by various names, including simple Bayes and independent Bayes . These names refer to Bayes’ theorem in the classifier’s decision rule, but a Naive Bayesian classifier is not necessarily a Bayesian method . Naive Bayes is a simple technique for constructing classifiers. A classifier is a model that assigns class labels to problem instances, represented as vectors of feature values, and the class labels are values from some finite set. The family of Naive Bayes algorithms is based on a common principle that the value of a particular feature, given a class variable, is independent of the value of any other feature. For example, a fruit can be considered an apple if it is red, round, and about ten centimeters in diameter. A Naive Bayesian classifier considers each feature to contribute to the probability that this fruit is an apple, regardless of any possible correlations between the color, shape, and diameter features.

In many practical applications, parameter estimation for Naive Bayes models uses the maximum likelihood method. In other words, one can work with a Naive Bayesian model without accepting Bayesian probability or methods. Despite their simple design and seemingly oversimplified assumptions, Naive Bayes classifiers have performed quite well in many complex real-world situations. In 2004, an analysis of the Bayesian classification problem showed that there are reasonable theoretical reasons for the seemingly incredible performance of Naive Bayesian classifiers . However, a comprehensive comparison with other classification algorithms in 2006 showed that Bayesian classification performed worse than other approaches, such as boosted trees or random forests . The advantage of Naive Bayes is that it requires only a small amount of training data to estimate the parameters needed for classification .

In the abstract, Naive Bayes is a conditional probability model. The model assigns probabilities to each of the possible outcomes or classes given the problem instance to be classified, represented by the vector which encodes features representing mutually independent variables . The problem with the above formulation is that if the number of features is large or the feature has a large value range, it is infeasible to base a model on probability tables. The model must therefore be reformulated to make it easier. The conditional probability can be decomposed using Bayes’ theorem in Equation [[eqn:5]](#eqn:5).

Using natural language and Bayesian probability terminology, the above equation can be written in a form presented in Equation [[eqn:6]](#eqn:6).

In practice, only the numerator is evaluated, because the denominator does not depend on , and the values of the feature are given. This means that the denominator is constant in practice. The numerator is equivalent to the pooled probability model which can be reformulated as follows, using the chain rule to repeatedly apply the definition of conditional probability in Equation [[eqn:7]](#eqn:7).

In the next step, "naive" assumptions about conditional independence are accounted for. Assume that all features in are mutually conditionally independent considering the category . Under this assumption, , and therefore the joint probability model can be expressed in Equation [[eqn:8]](#eqn:8)

The value indicates proportionality because the denominator is omitted. Under the above independence assumptions, the conditional distribution over the class variable is defined in Equation [[eqn:9]](#eqn:9).

The proof in Equation [[eqn:10]](#eqn:10) denotes a scaling factor that depends exclusively on , and constant if the feature variable values are known.

The discussion up to this point has been about the Naive Bayesian probability model with independent features. A Naive Bayesian classifier combines the specified model with decision rules. One frequently used decision rule is the selection of the most likely hypothesis to minimize the probability of misclassification, also known as maximum a posteriori (MAP). The corresponding classifier, the Bayesian classifier, is a function that assigns a class label to the sample for some class according to Equation [[eqn:11]](#eqn:11).

The a priori probability for each class can be calculated using Equation [[eqn:12]](#eqn:12) if we assume that the classes are equally likely, or by an estimate obtained by dividing the number of samples of the class by the total number of samples in the data set for training.

To estimate feature distribution parameters, we must assume a parametric distribution, or generate non-parametric models for the features from the training data set . Assumptions about the distribution of features are called an event model or Bayesian classifier, which can introduce confusion by creating both a continuous and discrete model . Multinomial and Bernoulli distributions have often been used for discrete features in document classification, or spam filtering. The Gaussian Naive Bayesian classifier was used since the features used in this study assumed continuous values.

When processing continuous data, it is often assumed that the values are distributed according to a normal (Gaussian) distribution. For example, suppose the training data contains a continuous attribute, . The data is first segmented according to the class, then according to the mean value and the variance of the variable which is calculated for each class separately. Let be the mean value of the sequence that denotes the class , and let be the Bessel corrected variance of the sequence that denotes the class . Suppose the collected observation value is . Then the probability density for the class , , can be calculated by including in the equation for the normal distribution parameterized by and . The expression in Equation [[eqn:13]](#eqn:13) is formally valid.

Clustering is often used to discretize continuous values. The new set of features obtained by discretization follows the Bernoulli distribution. Some literature claims that discretization is necessary to use a Naive Bayesian classifier, but discretization may discard the information needed to distinguish between classes .

Sometimes the distribution of class-based marginal densities is far from normal. The marginal probability density of each class can be used to estimate the kernel probability density function in such cases. This method, introduced by John and Langley , can significantly increase the accuracy of the classifier .

### Partial Least Squares

Partial Least Squares (PLS) regression, or projection to latent structures, , is a linear regression statistical model that transforms the predicted and the observable variables to a new space instead of identifying maximum variance hyperplanes used in a linear regression model. Svante and Herman O. A. Wold, a Swedish father and son, founded this approach. PLS methods are bilinear factor models because the and are projected to new spaces. In Partial Least Squares Discriminant Analysis (PLS-DA), is categorical .

PLS discovers how and are related, modeling their covariance structures with a latent variable approach. PLS determines the multidimensional direction in the space that explains the maximum multidimensional variance direction in the space. PLS regression is recommended when there are fewer observations than variables in the predictor matrix or multicollinearity is present among values. By contrast, standard regression will not succeed without regularization in such problems. PLS regression originated in social sciences but is widespread in chemometrics, bioinformatics, sensometrics, neuroscience, and anthropology.

Using paired observations . PLS finds the normalized direction that maximizes the covariance in the first step , shown in Equation [[eqn:14]](#eqn:14). Matrix notation is used in further discussion of the model.

The general underlying model of multivariate PLS with components is , and where is an matrix of predictors, and is an matrix of responses. and are matrix projections of (the score, component or factor matrix) and projections of (the scores), and are and loading matrices, and and are the error terms, assumed to be independent random normal variables following the same distribution.

The decompositions of and aim to increase the covariance between and . Covariance is defined pair by pair and the covariance of columns of equal length , of , and of , is maximized. The covariance of the column of with the column of is zero if does not equal . The scores form an orthogonal basis in PLSR, and the loadings are chosen to achieve this. This is a major difference with Principal Component Analysis (PCA) where orthogonality is imposed onto loadings (and not the scores) .

Many variants of PLS exist for estimating the factor and loading matrices , , , and . Most of them construct estimates of the linear regression between and as . Some PLS algorithms are only appropriate if is a column vector. Others deal with the general case of a matrix . Algorithms also differ in estimating the factor matrix as an orthogonal matrix. The final prediction will be the same for all these varieties of PLS, but the components will differ. PLS is composed of iteratively repeating the following steps times (for components): finding the directions of maximal covariance in input and output space performing least squares regression on the input score deflating the input , target , or both.

PLS1 is a widely used algorithm if is a vector that estimates as an orthonormal matrix. The vectors need to be normalized, which is not shown in the pseudocode in Table [2](#tab:PLS1). Variables in capital letters are matrices, variables in lower case letters are vectors if superscripted, and scalars if subscripted.

PLS1 pseudocode

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | **function** PLS1 | | | |
| 2 |  |  | | |
| 3 |  | , an initial estimate of . | | |
| 4 |  | **for** **to** | | |
| 5 |  |  |  | |
| 6 |  |  | (note this is a scalar) | |
| 7 |  |  |  | |
| 8 |  |  |  | |
| 9 |  |  | (note this is a scalar) | |
| 10 |  |  | **if** | |
| 11 |  |  |  | , **break** the **for loop** |
| 12 |  |  | **if** | |
| 13 |  |  |  |  |
| 14 |  |  |  |  |
| 15 |  | **end for** | | |
| 16 |  | **define** to be the matrix with columns | | |
|  |  | . | | |
|  |  | Do the same to form the matrix and vector. | | |
| 17 |  |  | | |
| 18 |  |  | | |
| 19 |  | **return** | | |

The input and are centered implicitly in the presented pseudocode. This pseudocode "deflates" the matrix (subtraction of ). Deflation of the vector is unneeded because it does not affect the result . The user-defined variable is the maximum number of latent factors in the regression. If equals the rank of the matrix , the result equals the least squares regression estimates for and .

### Flexible Discriminant Analysis

Flexible Discriminant Analysis (FDA) is a general methodology that creates the discriminant surface for a multigroup non-linear classification model based on a mixture of non-parametric linear regression models , such as Multivariate Adaptive Regression Splines (MARS) and linear discriminant analysis (LDA). FDA uses optimal scoring to transform the response variable to prepare the input data for linear separation. The range of predictor values is partitioned into several groups or categories using multiple regression models, called basis functions (BF). In the second step, LDA is run on the previously created classes to increase the variance between classes and decrease the variance within classes. Most of the variance between classes is explained by the first axis FDA creates. Each of the following predictors accounts for a smaller percentage of the variance between classes. This goes on until all of the variance between classes is accounted for. A -dimensional chart with two predictor pairs can be created to increase clarity.

Many predictors can be used at once in FDA , variable interactions are automatically noted , it is complex but execution time and computational load are adequate . The algorithm is not significantly affected by outliers .

Modifying different settings has a high impact on FDA , and the estimation might be unsuccessful if the predictor distributions are highly correlated. FDA is prone to overfitting , and challenging to comprehend or explain . The assumption of normality for the continuous independent variables is necessary to determine the response or grouping categorical variable.

In nonparametric regression analysis, the relationship between predictors and the dependent variable does not have a distribution with previously determined parameters. This function is created based on the input. Because of this additional step, it is necessary to gather more samples than for parametric estimation. For the input and output random variables and , we assume the relationship in Equation [[eqn:15]](#eqn:15) is true, where represents a deterministic function.

Linear regression is a restricted case of nonparametric regression where is assumed to be the composition of a linear function with a translation. A stronger assumption can be made using a random variable , representing added noise with a zero mean, where . We cannot achieve an unbiased estimate for without assuming it belong to a certain parametric distribution, suck as the normal or Gaussian distribution. Under unchanging conditions, these assumptions hold for most robust estimators.

Linear discriminant analysis (LDA), normal discriminant analysis (NDA), or discriminant function analysis is a generalization of Fisher’s linear discriminant defined in 1936 . LDA finds a linear combination of features differentiating two or more sample classes in statistics and other research. The features may be utilized directly or to reduce dimensionality before classification, which is more often used .

LDA is highly similar to the analysis of variance (ANOVA) and multivariate analysis of variance (ANOVA) methods and regression analysis. These methods express one dependent variable as a linear combination of other features or measurements . Discriminant analysis has continuous independent variables and a categorical dependent variable representing the groups . ANOVA and MANOVA use categorical independent variables and a continuous dependent variable. Because of these different variable types, ANOVA and LDA are less similar than LDA and logistic or probit regression. However, the LDA method assumes independent variables are normally distributed, which is not necessarily true. Logistic and probit regression explain a categorical variable by the values of continuous independent variables and do not make normality assumptions.

LDA is also closely related to PCA and factor analysis since they identify linear functions of multiple variables that fit the input with the smallest error . LDA depends on the group difference. However, PCA and factor analysis choose the model using similarities. Unlike factor analysis, discriminant analysis is not a standalone algorithm and depends on other methods for differentiating independent and dependent criterion variables.

LDA uses continuous values of observed independent variables. Discriminant correspondence analysis can similarly handle categorical independent variables. The groups are unknown in cluster analysis, which is not the case in discriminant analysis. Points in discriminant function analysis are rated based on quantitative predictors and a group measure to represent classification or distribution into groups or categories of the same type.

A class is known for a set of observations , also known as features, attributes, variables, or measurements for each sample of an object or event. This collection represents the training set in supervised learning. Finding a good predictor for the class , given only an observation for data other than the training set, but belonging to the same distribution, is a classic classification problem .

LDA assumes the conditional probability density functions and follow a normal distribution with mean and covariance parameters and , respectively. Equation [[eqn:16]](#eqn:16) shows that, if the log of the likelihood ratios is bigger than a threshold , the Bayes-optimal solution under the previous assumptions is to predict points as being from the second class.

Quadratic Discriminant Analysis (QDA) uses this classifier without additional assumptions. LDA assumes that the covariances have full rank and also assumes homoscedasticity, meaning that the class covariances are identical, as written in Equation [[eqn:17]](#eqn:17).

In this case, terms and cancel because is Hermitian and the above decision criterion becomes a threshold on the dot product for some threshold constant in Equation [[eqn:18]](#eqn:18), where .

Whether input belongs to a class depends on this linear combination of the measured observations. Using geometry, whether belongs to a class depends on a function of a projection of a multidimensional-space point onto s vector , so only the direction of the vector is considered. If the corresponding point is located on a certain side of a hyperplane perpendicular to , the observation belongs to . The location of the plane is defined by the threshold .

The assumptions of MANOVA are also true for discriminant analysis. The analysis is highly affected by outliers so there must be fewer predictor variables than samples in the smallest class . In every level of the grouping variable, independent variables must be normal to achieve multivariate normality . Box’s M statistic test whether homogeneity of variance/covariance is established and whether variances among group variables change for predictor levels.

Quadratic instead of linear discriminant analysis should be utilized if covariances are different . The scores of one predictor variable should not depend on other predictor variables for any sample, and the samples should be chosen at random . Discriminant analysis can handle small deviations from these requirements and may be used even if multivariate normality is not achieved, such as for dichotomous variables .

One or more linear combinations of predictors create a new latent variable for each discriminant function in discriminant analysis. The number of functions is the smaller possible value for , the number of groups , or the number of predictors . The first function increases the differences between groups for that function. The second function does the same but is not correlated with the first function. This is repeated for additional functions with the condition that the new function is uncorrelated with those defined before.

The discriminant rules suggest that if , then for a group , and sets of sample space. The discriminant analysis method identifies regions of that decrease errors of classification . A discriminant score describes how well it differentiates classes. The correlation between each discriminant score and predictor is represented by a structure correlation coefficient that is a zero-order correlation, meaning that it is not corrected for the other predictors .

Standardized coefficients represent each predictor’s weight in the linear combination forming the discriminant function. These coefficients are corrected for the other predictors and are partial, the same as for a regression equation. Standardized coefficients showcase the unique contribution of each predictor in determining class labels. Functions at group centroids are mean discriminant scores for grouping variables for each function. If the difference in means is large, the classification error is small.

The maximum likelihood rule increases class density when choosing the class of . The Bayes Discriminant Rule Assigns aims to increase the value of when choosing the class of , where is the prior probability of classification, and is the class density . Fisher’s linear discriminant rule aims to increase the ratio between and , identifying a linear combination of the predictors to determine labels .

The discriminant analysis eigenvalue represents the characteristic root of each function, indicating the quality of the group separation produced by the function. The function differentiates better for a larger eigenvalue . Eigenvalues do not have an upper limit , so the results should not be taken at face value. The eigenvalue can be viewed as a ratio of and as in ANOVA when the dependent variable is the discriminant function, and the groups are its levels . The largest eigenvalue corresponds to the first function, the second largest with the second, repeated for all following eigenvalues.

Even though it is not universally accepted, some suggest eigenvalues be utilized as effect size measures . Canonical correlation is a more established measure of effect size similar to the eigenvalue, representing the correlation between groups and the function , and equaling the square root of the ratio of and . The percent of variance can also be calculated for each function by , where is the eigenvalue for the function, and is the sum of all eigenvalues indicating the strength of the prediction for one particular function compared to the others . The percent of correctly classified data, such as the kappa value, can also be used as the effect size while accounting for random matches in classification . Kappa is not biased for any particular class, whether performance is high or low, and instead normalizes across all classes .

Fisher’s linear discriminant and LDA are sometimes considered synonyms. However, Fisher describes a slightly different discriminant, which does not make some of the assumptions of LDA such as a normal distribution or the equality of covariances for groups. The linear combination of features will have means and variances for if two classes of observations have means and covariances . Fisher uses the ratio of the variance between and within the classes to separate these two distributions in Equation [[eqn:19]](#eqn:19).

This measure can be considered a measure of the signal-to-noise ratio for the class labels. The maximum separation is achieved when . This above equation defines LDA, provided the assumptions made were true. The vector is the normal to the discriminant hyperplane. The separation line for a two-class task is perpendicular to , illustrating this claim.

The classified samples are projected onto and then a threshold dividing them is selected among one-dimensional distributions. No universal guidelines exist for this selection. An often-used selection is a hyperplane between projections of the two means, and , used if projections of data from both groups have similar distributions. Equation [[eqn:20]](#eqn:20) denotes how to find the parameter in the threshold condition for this hyperplane.

Otsu’s method, related to Fisher’s linear discriminant, chooses a black/white threshold that minimizes intra-class variance to convert a pixel histogram to grayscale and maximizes inter-class variance within/between pixels in grayscales assigned to black and white pixel classes. Canonical discriminant analysis (CDA) divides classes with the fewest errors by finding axes that define canonical coordinates. The -dimensional input is projected to a -dimensional space by using linear functions that are not correlated.

The application of the Fisher discriminant is expanded to find a subspace capturing all class variability for multiple groups, as suggested by C. R. Rao . Then The scatter between class variability is defined by the sample covariance of the class means in Equation [[eqn:21]](#eqn:21) if each of classes has a mean and the same covariance , where is the mean of the class means.

Equation [[eqn:22]](#eqn:22) describes class division in the direction .

The division will equal the eigenvalue when is an eigenvector of . The variability between features will be contained in the subspace defined by the eigenvectors of the largest eigenvalues if is diagonalizable, since the rank of is equal to or less than . PCA and other dimensionality reduction methods utilize these eigenvectors. Regularisation is recommended to reduce the impact of training data on eigenvectors corresponding to the smaller eigenvalues.

Alternative techniques are used for classification in place of dimensionality reduction. For instance, the classes may be partitioned, and a standard Fisher discriminant or LDA is used to classify each partition. LDA is applied in a "one against the rest" approach where the points from one class are juxtaposed with others. The results of classifiers are joined. In contrast, a new classifier is created for class pairs in pairwise classification.

### Principal Component Analysis

Principal component analysis (PCA) is a linear dimensionality reduction technique with applications in exploratory data analysis, visualization, and data preprocessing. It is used in preprocessing before training a Neural Network in the method in the *caret* package in *R* . The data is linearly transformed into a new coordinate system such that the directions (principal components) capturing the largest variation in the data can be easily identified.

The principal components of a collection of points in a real coordinate space are a sequence of unit vectors, where the -th vector is the direction of a line that best fits the data while being orthogonal to the first vectors. Here, a best-fitting line is defined as one that minimizes the average squared perpendicular distance from the points to the line. These directions (i.e., principal components) constitute an orthonormal basis in which different individual dimensions of the data are linearly uncorrelated. Many studies use the first two principal components to plot the data and visually identify clusters of closely related data points .

When performing PCA, the first principal component of a set of variables is the derived variable formed as a linear combination of the original variables that explains the most variance. The second principal component explains most of the variance once the first component is removed, repeated times until all the variance is explained. PCA reduces the number of highly correlated variables to an independent set. The first principal component can be defined as a direction with the largest variance of the projected data. The -th principal component has a direction orthogonal to the first principal components maximizing the projected data variance.

The principal components are eigenvectors of the data’s covariance matrix and are often computed by eigendecomposition of the data covariance matrix or singular value decomposition of the data matrix. PCA is closely related to factor analysis and is the simplest of the true eigenvector-based multivariate analyses. Factor analysis usually uses more domain-specific assumptions about the underlying structure and solves eigenvectors of a slightly different matrix. PCA is also related to canonical correlation analysis (CCA). PCA defines a new orthogonal coordinate system that optimally describes the variance in a single dataset . CCA does the same for the cross-covariance between two datasets. Robust and L1-norm-based variants of standard PCA have also been proposed .

Karl Pearson conceived of PCA in 1901 as a counterpart of the principal axis theorem in mechanics . Harold Hotelling discovered the Hotelling transform in multivariate quality control in the 1930s without knowledge of the previous work . Synonyms for PCA include the discrete proper orthogonal decomposition (POD) in mechanical engineering and the Kosambi–Karhunen–Loève or Karhunen–Loève theorem (KLT) in signal processing. Singular value decomposition (SVD) of X, established in the final quarter of the 19th century , is also similar to PCA. Eigenvalue decomposition (EVD) of a matrix in linear algebra shares the mathematical foundations of PCA . Factor analysis differs in several key features , but has many aspects in common with PCA. Empirical orthogonal functions (EOF) in meteorological science were developed by Lorenz in 1956 . The Eckart–Young theorem and quasiharmonic modes are both connected to PCA. Empirical modal analysis in structural dynamics and spectral decomposition in noise and vibration also take a similar approach to PCA.

PCA is similar to fitting a -dimensional ellipsoid to the data, with a principal component modeled by an ellipsoid axis. If the variance along an axis increases, the ellipsoid does as well. A variable must be centered by subtracting the mean to determine the ellipsoid axes. The covariance matrix of the data, eigenvalues, and eigenvectors are defined after this step. To obtain unit vectors, the orthogonal eigenvectors are normalized. The axes of the ellipsoid fitted to the data are then represented by the mutually orthogonal unit eigenvectors. The covariance is diagonalized by the selected basis, with the variance of each axis contained in the diagonal elements. If the eigenvector is divided by the eigenvalue, its proportion of the variance can be obtained. PCA analysis often includes biplots and scree plots, displaying the degree of explained variance. The start of the bend in a scree plot, the infection point or "knee", is a guideline for the number of components to keep.

The data in PCA is converted to a new coordinate system using a scalar projection, an orthogonal linear transformation on a real inner product space. The first coordinate or principal component covers the largest portion of the variance, decreasing with each following component . PCA uses an data matrix , where the columns representing features have been shifted so that the means equal zero, and each of the rows is an experiment iteration. The transformation given by Equation [[eqn:23]](#eqn:23) uses a set of of -dimensional vectors containing weights or coefficients and maps row vectors of to new vectors of principal component scores .

The variables of have the maximum possible variance, the same as for . The coefficient vectors are unit vectors. The number of features is commonly larger than to reduce dimensionality. The matrix equivalent of the previously stated claims is where , , and .

The first weight vector in Equation [[eqn:24]](#eqn:24) has maximum variance, with the matrix form in Equation [[eqn:25]](#eqn:25).

The vector needs to conform to Equation [[eqn:26]](#eqn:26) to be a unit vector.

The value we aim to increase can be considered a Rayleigh quotient. A positive semidefinite matrix, for example, , produces an output where the quotient’s maximum possible value is the largest eigenvalue of the matrix and is the corresponding eigenvector. The first principal component of a data vector equals in the original and in the transformed space, once is calculated.

By subtracting the first principal components from , we obtain tThe -th component in Equation [[eqn:27]](#eqn:27).

Equation [[eqn:28]](#eqn:28) describes using the new data matrix to determine the weight vector with the maximum variance. This weight vector represents the eigenvectors of that are still in consideration, with the maximum values for the quantity in brackets given by their corresponding eigenvalues.

If is the -th eigenvector of , the -th principal component of a data vector equals in the original and in the transformed space. If is a -by- matrix of weights and the eigenvectors of are columns, the complete principal components decomposition of is equal to . The whitening or sphering transformation uses the transpose of . Loadings in PCA or factor analysis equal the eigenvectors scaled up by variances, or the columns of multiplied by the square root of corresponding eigenvalues. The matrix is proportional to the empirical sample covariance matrix . Equation [[eqn:29]](#eqn:29) contains the sample covariance between two principal components.

When progressing from line to line , the eigenvalue property of is utilized. If the eigenvalues of a symmetric matrix are not the same, the eigenvectors and corresponding to them are orthogonal. If the vectors have the same repeated value, they can be orthogonalised. There is no sample covariance between principal components so the final product is zero. Principal components can also be described using transformation coordinates that convert the empirical sample covariance matrix to a diagonal matrix.

The empirical covariance matrix for the original variables in matrix form equals . If is the diagonal matrix of eigenvalues of , the empirical covariance matrix for the principal components is equal to . The sum of squares for a component is equal to , demonstrated in Equation [[eqn:30]](#eqn:30).

The transformation maps a data vector from an original to a new space of uncorrelated variables. If we retain only the first principal components defined by eigenvectors, the truncated transformation is given by the matrix that has only columns, but still has rows. PCA thus defines a linear transformation . The columns of A matrix form an orthogonal basis for the decorrelated features representing The transformed data matrices with only columns lower the squared reconstruction error or and increase the preserved variance.

This is useful for representing high-dimensional datasets while keeping most of the variance. Retaining the first two principal components facilitates the discovery of separated clusters in a two-dimensional plot. If data is randomized, clusters may overlap and cannot be identified. The chance of overfitting in regression analysis increases with the number of explanatory variables allowed, so conclusions cannot be extrapolated to different data. When possible explanatory variables are highly correlated, principal component regression proposes that they can be reduced to a few principal components against which the regression is run. Noisy data can also benefit from dimensionality reduction. Identically distributed Gaussian noise is present in the columns of if this was true for the original dataset. The Gaussian noise distribution is invariant if the is applied since it represents a rotation of the coordinate axes with a high dimensionality. A lower signal-to-noise ratio and a large noise impact are present in the later principal components containing a smaller part of the total variance. The largest part of the signal in PCA can be found in the first components. The other components may be eliminated with acceptable errors since they mostly contain noise. The significance of the principal components can be checked with parametric bootstrap for a smaller dataset to help decide how many principal components to keep .

## Data Description and Analysis

The samples are divided into training and testing datasets as close as possible to a ratio of for training and for testing. During the division, the share of classes in the original data was taken into account, that is, the division was stratified by class so that an approximately equal ratio of classes was present in both the training data and the testing data, which is a feature of the *createDataPartition* function from the *caret* R library that was used .

## Performance Metrics

### Confusion Matrix

The metrics and terminology used to evaluate the classifier performance were taken from the confusion matrix defined in the R function *confusionMatrix* in the *caret* library . The confusion matrix is suitable for use in multiclass classification, which is the goal of this project. A confusion matrix is a table where the number of rows and columns corresponds to the number of classes. The cells report the number of samples in each class classified into any class. This enables a more detailed analysis than by observing the share of correct classifications, i.e. accuracy. Accuracy will give results that can lead us to a wrong conclusion if the data set is unbalanced, that is, if the numbers of observations in different classes vary significantly, which was the case in this paper, because out of a total of samples, of them belonged to the largest N class, and to the smallest E class.

If there are only two classes (Yes and No), the metrics are calculated using an example of a confusion matrix in Table [3](#tab:cm).

A true positive (TP) classification result correctly indicates that the sample belongs to the positive class.

A true negative (TN) classification result correctly indicates that the sample belongs to the negative class.

A false positive (FP) classification result or type I error, incorrectly indicates that the sample belongs to the positive class because it truly belongs to the negative class.

A false negative (FN) classification result or type II error, incorrectly indicates that the sample belongs to the negative class because it truly belongs to the positive class.

The confusion matrix for a two-class problem.

|  | Reference | |
| --- | --- | --- |
| Prediction | Yes | No |
| Yes | TP | FP |
| No | FN | TN |

Sensitivity, recall, hit rate, or true positive rate (TPR), is calculated as , that is, the proportion of samples that are correctly classified as positive among all samples that truly belong to the positive class. Higher TPR values indicate better results for the positive class.

Specificity, selectivity, or true negative rate (TNR), is calculated using the expression , that is, the proportion of samples that are correctly classified as negative among all samples that truly belong to the negative class. Higher TNR values indicate better results for the negative class.

Prevalence is calculated as for the positive class, that is, the proportion of samples that truly belong to the positive class among all samples. Prevalence values other than an even split by the number of classes indicate a class imbalance.

Precision, or positive predictive value (PPV), is calculated as for a two-class problem, i.e. the proportion of truly positive results among all samples classified as positive. Higher PPV values indicate better results for the positive class.

The negative predictive value (NPV) is calculated as for a two-class problem, i.e. the share of truly negative results among all samples that are classified as negative. Higher NPV values indicate better results for the negative class.

The detection rate (DR) is calculated as for the positive class, that is, that is, the proportion of samples that are correctly classified as positive among all samples.

The detection prevalence (DP) is calculated as for the positive class, that is, that is, the proportion of samples that are correctly or incorrectly classified as positive among all samples.

The accuracy (Acc) is calculated as , that is, the proportion of samples that are correctly classified among all samples.

Since accuracy is not usable in the case of unbalanced classes, balanced accuracy (BA) is introduced, which is calculated as and is arithmetic mean TPR and TNR, which are respectively centered on the positive and on the negative class separately.

Another metric that can be used instead of accuracy to consider each class separately is the score, i.e. the harmonic mean of PPV and TPR, equal to .

If there are more than two classes, results are calculated for each class separately by viewing that class as the positive class and samples of all other classes as negative, a "one versus all" approach. The unweighted Kappa statistic and a *p*-value from McNemar’s test are also computed. McNemar’s test produces NA values with sparse tables, such as the one in this experiment, so it was not included in the results.

A confidence interval for the accuracy rate is computed using *binom.test* and a one-sided test to see if the accuracy is significantly larger than the "no information rate," which is the percentage of the largest class.

### The Binomial Test

The binomial test for two classes is an exact test of the statistical significance of deviations from a theoretically expected distribution.

The binomial test is often utilized to validate a hypothesis about the probability () of success , where users define between and . The expected number of successes would be if the null hypothesis were correct. If out of attempts are successful, the binomial distribution formula defines the probability in Equation [[eqn:31]](#eqn:31).

The probability of an outcome equally or more extreme than actual occurrences is used to calculate the -value. This is a simple operation for a one-tailed test. Equation [[eqn:32]](#eqn:32) contains the expression for the -value when testing the hypothesis that . We can use a range from to to test .

A binomial distribution isn’t symmetric if , so calculating a -value for a two-tailed test is not as simple. The -value from the one-tailed test cannot be doubled to obtain the result since all events must be considered, including ones more extreme than previously seen. These extreme events are equally or less probable than . Equation [[eqn:33]](#eqn:33) denotes all such events. Equation [[eqn:34]](#eqn:34) is used to compute the two-tailed -value.

### Cohen’s Kappa Coefficient

Cohen’s kappa coefficient (, lowercase Greek kappa) indicates inter- and intra-rater reliability for classification . As it accounts for random matches in ratings, it is a better metric than simply computing the percentage of agreement. Interpreting indices of agreement is not straightforward, and is still a point of contention among experts. Using disagreement instead of agreement in ratings might be simpler . Galton was the first to use a statistic similar to Cohen’s kappa in 1892 . Jacob Cohen formally described the kappa coefficient in 1960 in the journal "Educational and Psychological Measurement" .

Two raters group items into mutually exclusive classes, and Cohen’s kappa measures their agreement. Equation [[eqn:35]](#eqn:35) defines , where is the relative observed agreement among raters, and is the hypothetical probability of chance agreement. The observed data influences the computed probabilities of each observer randomly selecting each category.

If the raters completely align then . When there is no agreement other than what would be expected by chance and is given by , . If statistic is negative there is no match between the raters, which can reflect a real tendency of the raters to give differing ratings.

Equation [[eqn:36]](#eqn:36) is derived from the construction in Equation [[eqn:37]](#eqn:37) and holds for categories with observations to categorize if is the number of times rater predicted category . In these equations, is the estimated probability that both rater and rater will classify the same item as , and is the estimated probability that rater will classify an item as . A similar claim can be made for rater .

If ratings are independent Equation [[eqn:38]](#eqn:38), can be used.

Equation [[eqn:39]](#eqn:39) is used to estimate the term by using the number of items classified as by rater () divided by the total items to classify ().

(and similarly for rater ).

Equation [[eqn:40]](#eqn:40) contains the Cohen’s Kappa formula for the traditional confusion matrix for binary classifications in Table [3](#tab:cm).

In this scenario, Cohen’s Kappa equals the Heidke skill score introduced by Myrick Haskell Doolittle in 1888 .

# Research Results

The accuracy, CI, NIR, -value, and Kappa statistic for each model when using all variables (all), all variables except Dst (no Dst), all variables except Dst, TEC, and dTEC (no TEC), only , , and (coord), only , , and ( ), only , , and ( ), or only , , and ( ) is displayed in Table [4](#tab:stats:reverse:all), Table [5](#tab:stats:reverse:noDst), Table [6](#tab:stats:reverse:noTEC), Table [7](#tab:stats:reverse:coord), Table [8](#tab:stats:reverse:xyap), Table [9](#tab:stats:reverse:xzap), and Table [10](#tab:stats:reverse:yzap).

The accuracy, CI, NIR, -value, and Kappa statistic for each model when using all variables as input.

| Model | Acc | CI | NIR | -value | Kappa |
| --- | --- | --- | --- | --- | --- |
| svmPoly |  |  |  |  |  |
| C5.0 |  |  |  |  |  |
| nb |  |  |  |  |  |
| nnet |  |  |  |  |  |
| pls |  |  |  |  |  |
| fda |  |  |  |  |  |
| pcaNNet |  |  |  |  |  |

The accuracy, CI, NIR, -value, and Kappa statistic for each model when using all variables except Dst as input.

| Model | Acc | CI | NIR | -value | Kappa |
| --- | --- | --- | --- | --- | --- |
| svmPoly |  |  |  |  |  |
| C5.0 |  |  |  |  |  |
| nb |  |  |  |  |  |
| nnet |  |  |  |  |  |
| pls |  |  |  |  |  |
| fda |  |  |  |  |  |
| pcaNNet |  |  |  |  |  |

The accuracy, CI, NIR, -value, and Kappa statistic for each model when using all variables except Dst, TEC, and dTEC as input.

| Model | Acc | CI | NIR | -value | Kappa |
| --- | --- | --- | --- | --- | --- |
| svmPoly |  |  |  |  |  |
| C5.0 |  |  |  |  |  |
| nb |  |  |  |  |  |
| nnet |  |  |  |  |  |
| pls |  |  |  |  |  |
| fda |  |  |  |  |  |
| pcaNNet |  |  |  |  |  |

The accuracy, CI, NIR, -value, and Kappa statistic for each model when using only , , and as input.

| Model | Acc | CI | NIR | -value | Kappa |
| --- | --- | --- | --- | --- | --- |
| svmPoly |  |  |  |  |  |
| C5.0 |  |  |  |  |  |
| nb |  |  |  |  |  |
| nnet |  |  |  |  |  |
| pls |  |  |  |  |  |
| fda |  |  |  |  |  |
| pcaNNet |  |  |  |  |  |

The accuracy, CI, NIR, -value, and Kappa statistic for each model when using only , , and as input.

| Model | Acc | CI | NIR | -value | Kappa |
| --- | --- | --- | --- | --- | --- |
| svmPoly |  |  |  |  |  |
| C5.0 |  |  |  |  |  |
| nb |  |  |  |  |  |
| nnet |  |  |  |  |  |
| pls |  |  |  |  |  |
| fda |  |  |  |  |  |
| pcaNNet |  |  |  |  |  |

The accuracy, CI, NIR, -value, and Kappa statistic for each model when using only , , and as input.

| Model | Acc | CI | NIR | -value | Kappa |
| --- | --- | --- | --- | --- | --- |
| svmPoly |  |  |  |  |  |
| C5.0 |  |  |  |  |  |
| nb |  |  |  |  |  |
| nnet |  |  |  |  |  |
| pls |  |  |  |  |  |
| fda |  |  |  |  |  |
| pcaNNet |  |  |  |  |  |

The accuracy, CI, NIR, -value, and Kappa statistic for each model when using only , , and as input.

| Model | Acc | CI | NIR | -value | Kappa |
| --- | --- | --- | --- | --- | --- |
| svmPoly |  |  |  |  |  |
| C5.0 |  |  |  |  |  |
| nb |  |  |  |  |  |
| nnet |  |  |  |  |  |
| pls |  |  |  |  |  |
| fda |  |  |  |  |  |
| pcaNNet |  |  |  |  |  |

The execution time in seconds calculated using the R *system.time* function for each model when using different combinations of variables as input is displayed in Table [11](#tab:time:total). The experiment was run on Windows 11 using R Studio version 2024.04.2+764 and R version 4.4.1, the AMD Radeon RX 6600 GPU, 16GB of RAM, and the AMD Ryzen 5 PRO 4650G CPU with 6 cores.

The execution time in seconds for each model when using different combinations of variables as input.

| Model | svmPoly | C5.0 | nb | nnet | pls | fda | pcaNNet |
| --- | --- | --- | --- | --- | --- | --- | --- |
| all |  |  |  |  |  |  |  |
| no Dst |  |  |  |  |  |  |  |
| no TEC |  |  |  |  |  |  |  |
| coord |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

## A Comprehensive Analysis of the Naive Bayes Model

The confusion matrix and the performance indicators derived from it when using the Naive Bayes model and all input variables are depicted in Table [12](#tab:cm:all:nb) and Table [13](#tab:cs:reverse:all:nb). The same values when using all input variables except Dst are marked in Table [14](#tab:cm:noDst:nb) and Table [15](#tab:cs:reverse:noDst:nb), and Table [16](#tab:cm:yzap:nb) and Table [17](#tab:cs:reverse:yzap:nb) when using only , , and variables as input. These results are displayed since the Naive Bayes model has low computational load and high accuracy. The combinations of input variables that produced a accuracy when used for the Naive Bayes model were not included since there are no errors in classification.

The confusion matrix for the Naive Bayes model when using all variables as input.

|  | Reference | | | | |
| --- | --- | --- | --- | --- | --- |
| Prediction | E | N | P | R | T |
| E |  |  |  |  |  |
| N |  |  |  |  |  |
| P |  |  |  |  |  |
| R |  |  |  |  |  |
| T |  |  |  |  |  |

The performance indicators derived from the confusion matrix for the Naive Bayes model when using all variables as input.

|  | Class | | | | |
| --- | --- | --- | --- | --- | --- |
| Statistics | E | N | P | R | T |
| Sensitivity |  |  |  |  |  |
| Specificity |  |  |  |  |  |
| PPV |  |  |  |  |  |
| NPV |  |  |  |  |  |
| DR |  |  |  |  |  |
| DP |  |  |  |  |  |
| BA |  |  |  |  |  |

The confusion matrix for the Naive Bayes model when using all variables except Dst as input.

|  | Reference | | | | |
| --- | --- | --- | --- | --- | --- |
| Prediction | E | N | P | R | T |
| E |  |  |  |  |  |
| N |  |  |  |  |  |
| P |  |  |  |  |  |
| R |  |  |  |  |  |
| T |  |  |  |  |  |

The performance indicators derived from the confusion matrix for the Naive Bayes model when using all variables except Dst as input.

|  | Class | | | | |
| --- | --- | --- | --- | --- | --- |
| Statistics | E | N | P | R | T |
| Sensitivity |  |  |  |  |  |
| Specificity |  |  |  |  |  |
| PPV |  |  |  |  |  |
| NPV |  |  |  |  |  |
| DR |  |  |  |  |  |
| DP |  |  |  |  |  |
| BA |  |  |  |  |  |

The confusion matrix for the Naive Bayes model when using only , , and as input.

|  | Reference | | | | |
| --- | --- | --- | --- | --- | --- |
| Prediction | E | N | P | R | T |
| E |  |  |  |  |  |
| N |  |  |  |  |  |
| P |  |  |  |  |  |
| R |  |  |  |  |  |
| T |  |  |  |  |  |

The performance indicators derived from the confusion matrix for the Naive Bayes model when using only , , and as input.

|  | Class | | | | |
| --- | --- | --- | --- | --- | --- |
| Statistics | E | N | P | R | T |
| Sensitivity |  |  |  |  |  |
| Specificity |  |  |  |  |  |
| PPV |  |  |  |  |  |
| NPV |  |  |  |  |  |
| DR |  |  |  |  |  |
| DP |  |  |  |  |  |
| BA |  |  |  |  |  |

# Discussion

The Neural Network model has the highest execution time for any subset of the input variables due to its complexity and extensive training, evident from the data in Table [11](#tab:time:total). However, in each case, it achieved a accuracy, which is shown in Table [4](#tab:stats:reverse:all), Table [5](#tab:stats:reverse:noDst), Table [6](#tab:stats:reverse:noTEC), Table [7](#tab:stats:reverse:coord), Table [8](#tab:stats:reverse:xyap), Table [9](#tab:stats:reverse:xzap), and Table [10](#tab:stats:reverse:yzap). The Naive Bayes model has the second highest accuracy, over . The Naive Bayes model achieved a accuracy, except when using all input variables, all input variables except Dst, or only , , and . The difference in accuracy between the Neural Network and the Naive Bayes model is negligible, and the training time for the Neural Network model is more than twice as long. Despite their simple design and seemingly oversimplified assumptions, Naive Bayes classifiers have performed quite well in many complex real-world situations. In 2004., an analysis of the Bayesian classification problem showed that there are reasonable theoretical reasons for the seemingly incredible performance of Naive Bayesian classifiers .

All other models fail to achieve an accuracy over , except when using all input variables, including Dst. The Dst class was derived by thresholding the continuous Dst value and discretizing it by converting it to a single character. This labeling method explains the increase in accuracy when adding the Dst input variable in all except the two best-performing models. For example, the C5.0 model using Decision Trees did not consider any variable except Dst when included in the input, indicated by a variable importance. The svmPoly model is consistently the worst-performing for any subset of input variables, never achieving an accuracy over . None of the models achieved an accuracy under , so they were all moderately successful.

When studying the performance of the Naive Bayes model by individual class in Table [12](#tab:cm:all:nb), Table [13](#tab:cs:reverse:all:nb), Table [14](#tab:cm:noDst:nb), Table [15](#tab:cs:reverse:noDst:nb), Table [16](#tab:cm:yzap:nb) and Table [17](#tab:cs:reverse:yzap:nb), samples of the T class are all correctly classified, but the testing is less extensive since it is the second smallest class. Samples of the R class are sometimes erroneously assigned to the N, P, or less commonly the T class. Samples of the P class are rarely mistaken for the N class. Two of the most common misclassifications were assigning samples of the N class to the P or R classes. This is due to the largest number of samples in the N class between the P and R classes with the narrowest Dst range. Samples of the N class are least often included in the E class, whose range is the furthest apart from the N class. A sample of the E class was attributed to the N class only once. The E class is the smallest with only three samples, which must be accounted for when interpreting these results.

# Conclusion

The results indicate that even a limited subset of variables, including coordinates ad , available on simple devices such as smartphones, without measuring TEC and calculating Dst, can be successfully utilized to construct a simple and efficient model using a Naive Bayes classifier to determine Dst classes. The Neural Network model was marginally more accurate. However, the training time and model size were inappropriate for developing a new model with additional data or utilizing the model in practice.

# Declaration of competing interest

The authors declare no conflict of interest.

# CRediT authorship contribution statement

**Lucija Žužić:** Conceptualization, Methodology, Software, Validation, Investigation, Data Curation, Writing – Original Draft, Visualization.

**Deni Klen:** Conceptualization, Methodology, Software, Validation, Investigation, Data Curation, Writing – Original Draft, Visualization.

**Teodor B. Iliev:** Conceptualization, Validation, Formal analysis, Investigation, Resources, Writing – Original Draft, Writing – Review & Editing, Supervision.

**Renato Filjar:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Resources, Writing – Original Draft, Writing – Review & Editing, Supervision, Project administration, Funding acquisition.

# Funding

This work was fully supported by