Evaluation of Machine Learning Algorithms for Multiclass Classification of Voice Calls from Power Systems Operations

Matheus N. S. M. de Lima
DAE - Departamento Acadêmico de Eletrotécnica
IFSC - Instituto Federal de Santa Catarina
Florianópolis, Brasil
0000-0002-8435-2104

Fabrício Y. K. Takigawa
DAE - Departamento Acadêmico de Eletrotécnica
IFSC - Instituto Federal de Santa Catarina
Florianópolis, Brasil
0000-0002-1541-0518

***Abstract*—The Brazilian electric system is operated by a non-profit institution called the National Electric System Operator (ONS) which coordinates and controls the operation of energy generation and transmission. For this, the main form of operation occurs through verbal communication between the operators of the system. All telephone calls made between operators are recorded and stored on a server. In this sense, from the transcription and labeling of the audios, it is possible to use Natural Language Processing (NLP) and Machine Learning (ML) techniques to classify verbal communication in the operation. The Python programming language will be used together with a scikit-learn library to perform a comparison of multiclass classifiers, Complement Naive Bayes, Linear Support Vector Classification, Stochastic Gradient Descent Classifier, K-Nearest Neighbors Classifier, Multi-Layer Perceptron Classifier and Random Forest Classifier by tunning the hyperparameters and applying evaluation metrics for classification models.**

***Keywords*— Dataset, text, scikit-learn, classification, machine learning**

I. INTRODUCTION

Brazil has a National Interconnected System (SIN) with hundreds of thousands of kilometers of transmission lines and several sources of generation spread throughout its immense territory. This system is operated centrally by the National Electric System Operator (ONS), which makes it possible to optimize the country's energy resources by reducing operating costs both now and in the future [1].

Currently, the main means of communication between ONS operators and agents of the electricity sector is through telephone calls. All verbal communication carried out by the operators is recorded and stored on a server for later consultation by the post-operation sector, which is responsible for verification of the operation.

Electric sector operators follow the guidelines of the Grid Procedures (PR), which are documents that seek to standardize all activities carried out by ONS. In this context, it is worth mentioning the operation routine 4.7.1 of Sub-module 10.22 that standardizes Verbal Communication in Operation [2]. In the standardization, the dialogues between operators must follow a pattern of presentation, command, repetition of the command and confirmation.

The post-operation processes consist of listening to the communication calls between the operators and accessing the audio files to analyze and verify the activities. These consultations are time-consuming because they do not have an indication of the subject being discussed. To increase the efficiency of the post-operation analysis, data extraction and analysis techniques can be applied along with machine learning to classify the communication performed in the operation of the system.

For the execution of this study, a flowchart of the developed methodology was elaborated. Fig. 1 shows the steps adopted for the analysis of the estimators.

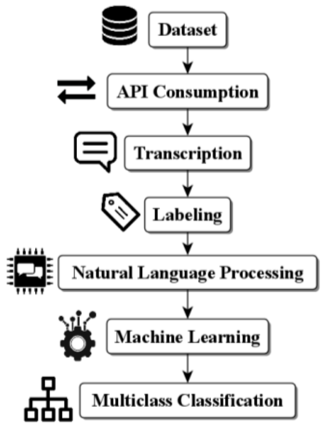


Fig. 1. General process adopted for data processing and analysis.

The dataset used consists of telephone calls, in WAV format, with a single channel and a sampling rate of 8000kHz, containing a dialogue related to the operation of the electricity sector.

The audio transcription was performed using an Application Programming Interface (API). In [3], the precision of the services IBM [4], Google [5] and Wit [6] is compared and it is concluded that Google presented the best performance, followed by Wit. For this study, the service chosen was Wit over Google due to the restriction imposed by the latter, namely a maximum of 60 minutes of transcription per month compared to no limits imposed by Wit.

After transcription of the audios, manual labeling was performed. Each sample received a category related to the subject of the telephone calls. Then proceed to the text processing step using Natural Language Processing (NLP) techniques to later applying supervised Machine Learning (ML) algorithms of the multiclass classification type.

In this study, the text samples related to operation of the electricity sector were transcribed, processed and analyzed using Python together with its libraries. The main problem was the multiclass classification of the samples.

The manuscript is organized as follows: Section 2 illustrates the necessary requirements for the development of the analysis and the methods used. In Section 3, the results are presented and the final considerations and proposals for improvement are outlined in Section 4.

II. MATERIAL AND METHODS

The study was developed using mainly the Python programming language as it is open-source, works on different operating systems, allows the use of several libraries made available by the community, has memory management, executes multiple threads and has a solid basic documentation [7]. For programming with Python, three items are needed:

- Core: language interpreter along with some standard libraries.
- Integrated Development Environment (IDE): interface used for development (Pyzo, PyCharm, Spider or Jupiter).
- Libraries: collection of script modules that perform a given activity.

For the processing and transcription of audios with Wit, the following libraries were explored:

- SpeechRecognition: speech recognition library, with support for various engines and APIs, online and offline.
- wave: provides an interface to the WAV sound format.
- auditok: library for detecting audio activity.
- glob: module to find all the names of a given path.
- jiwer: Python package for calculating Word Error Rate (WER), Match Error Rate (MER), Word Information Lost (WIL) and Word Information Preserved (WIP) of a transcript [8].
- pydub: library to manipulate audio with a simple and high-level interface.
- pathlib: library to handle file system paths independently.

For the pre-processing of transcriptions, there are libraries that perform this task automatically, such as NLTK in Python [9]. However, as the context is specific to electrical engineering and in Brazilian Portuguese, it was decided to perform the processing manually using the resources provided by the Python language. The following NLP techniques were applied [10]:

- removal of accentuation and punctuation (e.g., “Três” becomes “Tres”).
- conversion of corpus into lowercase letters (e.g., “Tres” becomes “tres”).
- plural removal (e.g., “unidades” becomes “unidade”).
- removal of diminutives and augmentatives (e.g., “pouquinho” becomes “pouco”).
- stop-words removal (e.g., “a”, “o”, “ao”, “de”, “da”, “em”).
- removal of words shorter than three characters (with the exception of “kv” and “mw”).
- correction of wrongly transcribed words (“caveira” becomes “kv”).
- lemmatization, bending the word for your lexeme (“limitada”, “limitado”, “limitava” becomes “limita”) or
- stemming, bending the word to its root (“convertido”, “converti”, “convertemos” become “convert”).

Regarding the analysis and application of ML techniques, it was decided to use the Python library and scikit-learn [11]. This and other useful libraries are listed below:

- scikit-learn: open-source machine learning library.
- numpy: package to support multidimensional arrays.
- pandas: similar to numpy, used for data analysis and manipulation.
- matplotlib: comprehensive library for data visualization.

A. *Labeling the samples*

The 1500 textual samples were labelled manually and were separated into twelve categories. The categories and the number of samples per category are listed in Table 1.

TABLE I. CLASSES AND QUANTITY OF SAMPLES.

Class	*Quantity*	*Class*	*Quantity*
load	51	supervision failure	53
proof of availability	52	hydrology	70
generation control	339	time confirmation	124
voltage control	358	no data	118
transmission control	59	interventions	106
converter	54	test communication	116

B. *Processing of the dataset*

To use the textual dataset in the estimators, it is necessary to convert the text documents into a numerical matrix of word frequency, called bag of words. To perform this task, scikit-learn provides the CountVectorizer function. After this conversion, the data is suitable for training the models. However, this method alone ends up ignoring less frequent words that could improve the training process of the models.

In this sense, it is recommended to use the statistical measure Frequency of Term (TF) and Inverse of Frequency in Documents (IDF) to consider the general weight of the word in relation to the document as a whole. The TfidfTransformer function of scikit-learn performs this task from the numerical frequency matrix of words. Fig. 2 shows the complete process of handling and processing textual samples.

Another way to add more value to the data is to group one and two (or more) words at the same time called n-grams. In this way it is possible to take into account the sequence of words together with unique words [12]. Scikit-learn enables this by changing a parameter of the TfidfVectorizer function, which performs CountVectorizer and TfidfTransformer together. For the proposed case, unigrams and bigrams were used.

After having finished processing the input data of the models, it was possible to select estimators according to the affinity with the proposed problem (multiclass textual classification). The six models chosen for evaluation and comparison were:

I. Complement Naive Bayes (ComplementNB) [13].
II. Linear Support Vector Classification (LinearSVC) [14].
III. Stochastic Gradient Descent Classifier (SGDClassifier) [15].
IV. K-Nearest Neighbors Classifier (KNeighborsClassifier) [16].
V. Multi-Layer Perceptron Classifier (MLPClassifier) [17].

Random Forest Classifier (RandomForestClassifier) [18].

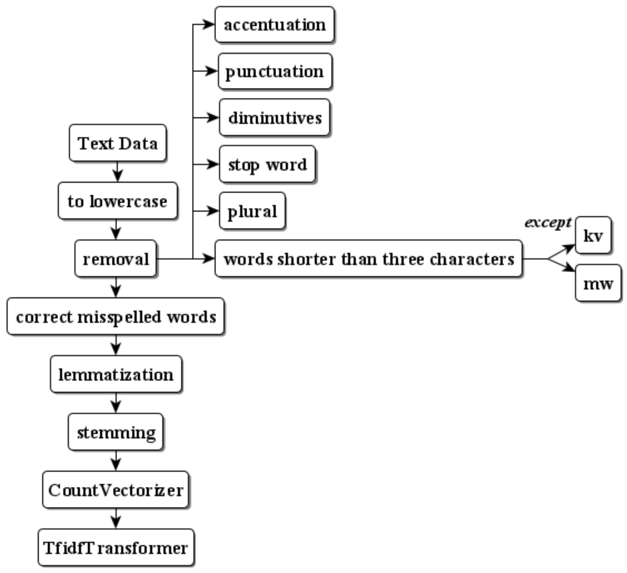


Fig. 2. Complete process of text handling.

C. *Complement Naive Bayes*

The Naive Bayes method is widely accepted since it presents a quick quality solution to textual classification problems [19]. In this sense, after its popularization, other variations were implemented seeking more attractive results. The ComplementNB [13] emerges as an improvement on the traditional Multinomial Naive Bayes, to compete with other famous methods like Support Vector Machines (SVM).

The ComplementNB corrects poor assumptions accepted by the traditional model exposed in [19], as it considers the non-uniformity of the data, the normalization of the weights, among others. These additional considerations raise the quality of the model [13].

D. *Linear Support Vector Classification*

This model implemented in scikit-learn in terms of the LIBLINEAR library proposed by [14], which is derived from the LIBSVM library proposed by [20]. LIBLINEAR, in relation to LIBSVM, adds greater flexibility in the choice of penalties and in the loss function. It is recommended to solve problems with large numbers of samples.

LinearSVC solves an optimization problem with different loss functions. As the SVM method natively only supports binary problems, the method uses the one-vs-the-rest strategy for multiclass problems [21] in which a binary classification problem is divided by class.

E. *Stochastic Gradient Descent Classifier*

Stochastic Gradient Descent (SGD), originally proposed by [22], is an efficient optimization method that seeks the linear learning of classifiers and regressors on convex loss functions such as the previously cited SVM and Logistic Regression. The SGD is comprehensive (can be used in deep learning) and is recommended for problems with large numbers of samples [23].

The SGDClassifier is a linear classifier (similar to the LinearSVC) optimized by the SGD method which also follows the one-vs-the-rest strategy. This more comprehensive model, compared to other estimators, allows the editing of more parameters, making it more complex to use. Depending on the choice of the “loss” parameter, Linear SVM, Logistic Regression or Probabilistic Logistic Regression Classifier will be implemented.

F. *K-Nearest Neighbors Classifier*

The KNeighborsClassifier is different from the others models, because does not attempt to construct a general internal model, it simply stores instances of the training data and the input is assigned to the class based in a determined number of neighbors. According with [16], comparing this estimator with the SVM, if the training data is much larger than the number of features, KNeighborsClassifier is considered better than SVM.

G. *Multi-Layer Perceptron Classifier*

MLPClassifier is a supervised learning algorithm that learns based in a number of dimensions for input and output the training occurs using backpropagation. Given a set of features it can learn a non-linear function approximator for either classification or regression. Between the input and the output layer, there can be one or more non-linear layers, called hidden layers [24].

Perceptron receives inputs, multiplies them by weights, and then passes them into an activation function (logistic, relu, tanh, identity) to produce an output. MLPClassifier trains iteratively since at each time step the partial derivatives of the loss function with respect to the model parameters are computed to update the parameters [17].

H. *Random Forest Classifier*

RandomForestClassifier is an ensemble-based method that fits a collection of decision tree classifiers and uses averaging to improve the predictive accuracy and control over-fitting. By default, the whole dataset is used to build each tree. Ensembles are often much more accurate than the individual classifiers that make them up [18-25].

I. *Training and testing*

Training and testing can be conducted in several ways. Initially, the dataset should be divided into two groups: training and testing. For this study, 80% of the samples were chosen for training and 20% for testing also called holdout set.

The training group is used to tuning the hyperparameters for the specific problem. To achieve the best set of hyperparameters to each model an exhaustive grid search. The library scikit-learn provided the GridSearchCV that performs this task, it fits the model using the training dataset for all the possible combinations of parameter values and evaluate to retain the best combination using Cross-Validation (CV) [26-27].

Common methods for performing CV are Kfolds and ShuffleSplit [11] (both are available in the stratified form, meaning that the partitions are selected so that the distribution of classes are approximately equal):

- Kfolds: divides the dataset into a specific number of folds in which each sample appears only once per fold and generates the samples for training and testing.
- ShuffleSplit: divides the dataset into a specified number of times and for each division selects samples at random to generate the samples for training and testing.

The Stratified Kfolds was used because ShuffleSplit is recommended for larges datasets to reduce fitting time and the present dataset is small and unbalanced [11].

III. Experiments and Results

A. *Setup of the estimators*

With the GridSearchCV combined with CV using five Stratified Kfolds the tunning hyperparameters was performed. An exhaustive search is performed to find the best combination of hyperparameters using the mean test F1 weighted of the folds as metric [26].

Mainly a text feature extraction was performed, to find what combination of vocabulary size (considering the most frequent words) and n-gram range brings the most accurate model. The Fig. 4 shows the average values of accurate for together with an important hyperparameter in the axes x (for this model the hyperparameter selected is alpha) related with the text extraction (by color indicating ngram range and shape related to the max features). Following the legend of the Fig 3.

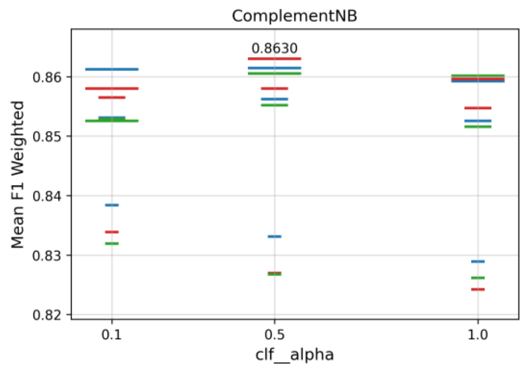


Fig. 3. Text feature extraction for ComplementNB.

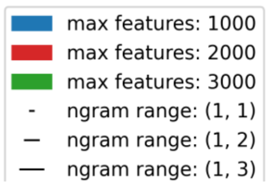


Fig. 4. Legend of text feature extraction.

The Fig. 3 shows that the model performed worst if the ngram range is set to (1, 1) and the best max feature vary depending of the ngram range and alpha. However, the best combination is alpha 0.5, max featured 2000 and ngram range (1, 3). The feature extraction of the remaining models are displayed by the Fig. 5 to 9. For each model the axis x represents an important hyperparameter and vary by the model type.

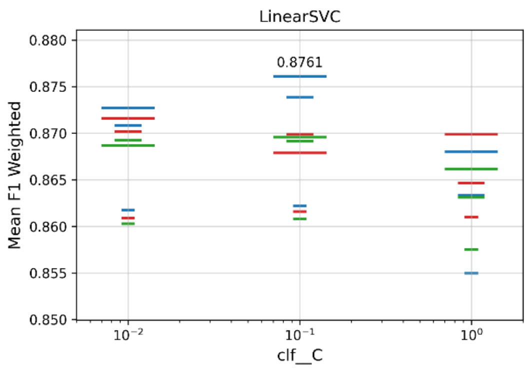


Fig. 5. Text feature extraction for LinearSVC.

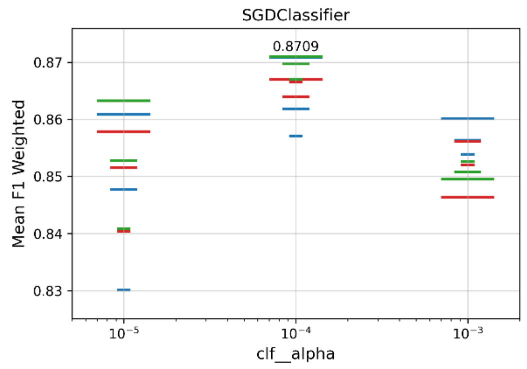


Fig. 6. Text feature extraction for SGDClassifier.

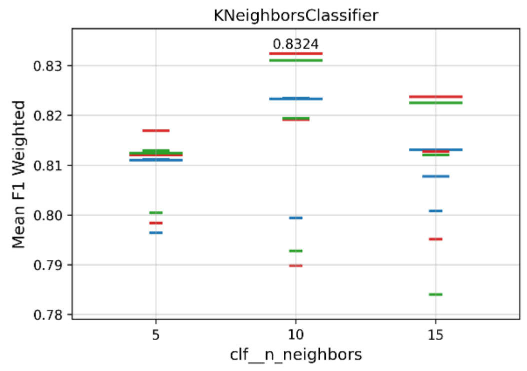


Fig. 7. Text feature extraction for KNeighborsClassifier.

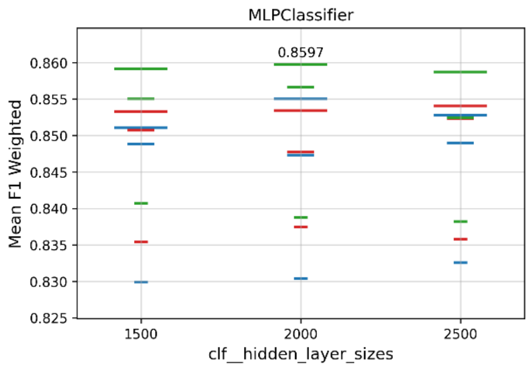


Fig. 8. Text feature extraction for MLPClassifier.

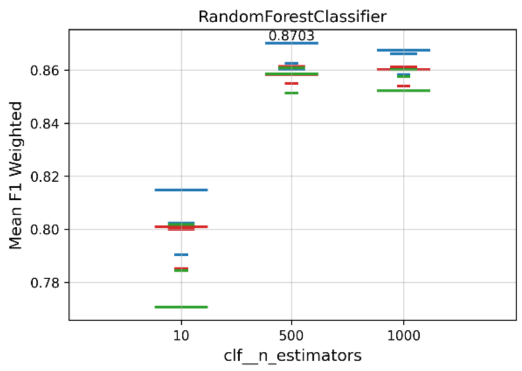


Fig. 9. Text feature extraction for RandomForestClassifier.

Observing the figures above it is possible to conclude, in relation with the text feature, that the best ngram range is (1, 3) for all estimators. Related to max features, consider the top words ordered by term frequency across the corpus, two models performed better with 1000 (LinearSVC and RandomForestClassifier), two with 2000 (ComplementNB and KNeighborsClassifier) and two with 3000 (SGDClassifier and MLPClassifier).

The setup of the estimators occurred following the best value of mean F1 weighted, searching for the best parameters related to text feature and for others specific parameters of each model (Alpha, C, Number of Neighbors, Hidden Layer Sizes and Number of Estimators).

B. *Results*

After the hyperparameter tunning, the models are able to fit the entire training set and predict the test set (holdout set) to evaluation. To perform the evaluation of each model, it is used the metrics [28]:

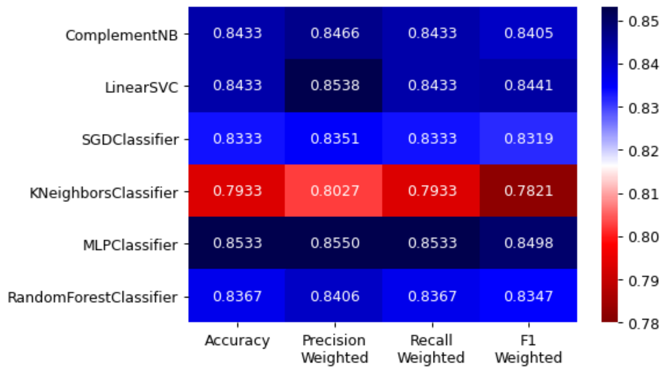
- Accuracy Score: fraction of correctly classified samples.
- Precision Score: of the total samples predicted to one class return how many are correct.
- Recall Score: of the total samples in one class return how many are predicted correct.
- F1 Score: it is interpreted as a weighted average of the precision and recall.

For this multiclass classification it is computed the mean weighted precision, recall and F1 score because the dataset is unbalanced, so all classes have the same importance, finding the overall performance of the model. The Fig. 10 shows the evaluation of each model for the holdout set. The table reveals that, for the metrics explored and the test set, the MLPClassifier is the most, most precise, higher recall and has the higher value of F1. As expected, the KNeighborsClassifier occupied the final position for this evaluation.

It is interesting to observe that RandonForestClassifier is a powerful estimator and for 500 estimators the MLPClassifier outperformed it, however there is a possibility of increase the number of estimators (losing computational time) for better results.

C. *Other results*

Using the precision, recall and F1 score of all models it is possible to evaluate how difficult is the prediction of each class. The Fig 10 shows the values of the metrics for each model and class and it is possible to observe that the easiest classes to predict are “proof of availability”, “converter”, “hydrology” and “communication test”. The hardest classes are “load”, “transmission control”, “supervision failure” and “time confirmation”



	Accuracy	Precision Weighted	Recall Weighted	F1 Weighted
ComplementNB	0.8433	0.8466	0.8433	0.8405
LinearSVC	0.8433	0.8538	0.8433	0.8441
SGDClassifier	0.8333	0.8351	0.8333	0.8319
KNeighborsClassifier	0.7933	0.8027	0.7933	0.7821
MLPClassifier	0.8533	0.8550	0.8533	0.8498
RandomForestClassifier	0.8367	0.8406	0.8367	0.8347

Fig. 10. Hold-out evaluation of each estimator.

From metric F1 in the Fig. 11, it is possible to conclude that all models performed well for some classes (the worst model is KNeighborsClassifier). The top model for “load”, “transmission control” is RandonForestClassifier, for “supervision failure” is SGDClassifier and “time confirmation” and “generation control” is MLPClassifier.

It is noted that some classes are very distinct than others, the easiest classes have words that do not repeat in other classes and the hardest classes have commons words. For example, it is possible that time confirmation happens in one sample of transmission control and load. Because of this, as elucidate in the final considerations, for future works is recommended the relabeling of the dataset using multilabel approach.

Another reason to the low performance of some classes is the quantity of samples in the holdout set. The nature of the problem is unbalanced, and of the 1500 labeled samples, 1/5 is separated for model evaluation. Some classes are less representative, “load”, “proof of availability”, “transmission control”, “converter” and “supervision failure” has less than 60 samples, leaving only 12 samples to evaluate. Therefore, it is possible that this low frequency of sample doesn’t truly represent the class so it is recommended more data for future works.

IV. FINAL CONSIDERATIONS

After the application of training and tests to the dataset for the selected models, is observed that the model MLPClassifier has the best overall performance. Related to the text feature extraction, the best results are observed when the N-gram range is (1, 3). When analyzing the average F1 score per class, some classes are easier to predict correctly than others. Another model that presents interesting results is LinearSVC.

It is possible to affirm that the classification of the verbal communication in the operation of the electric sector was performed successfully. For future work, it is recommended that the number of samples be increased because the selected estimators are able to work with higher quantities than the one used in this study and to prevent underfitting.

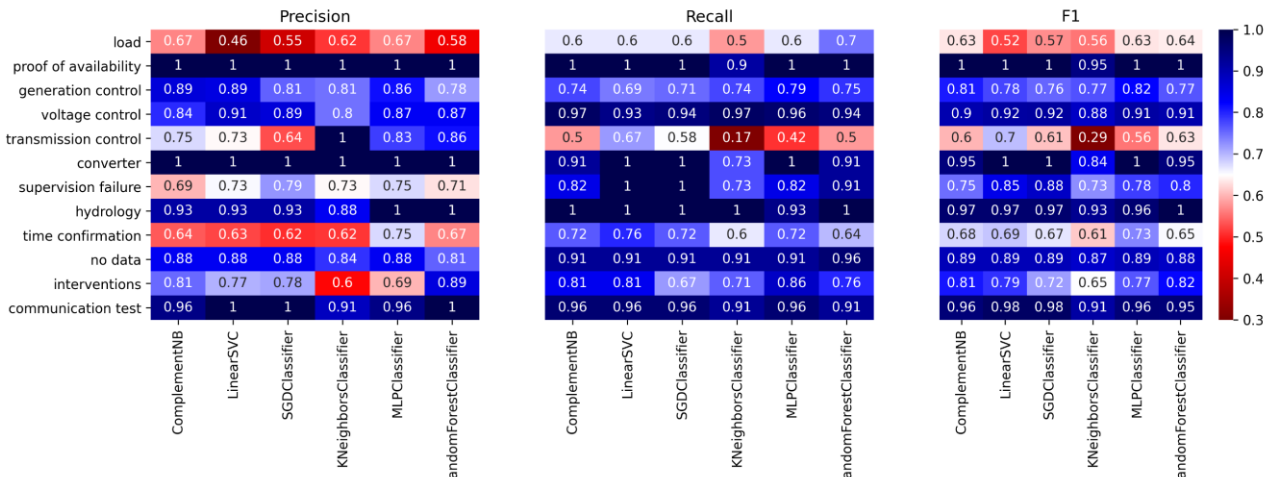


Fig. 11. Precision, Recall and F1 of all models per class.

References

[1] ONS, Resultado da apuração do PPR, 2019. Accessed on: Mar 07, 2021. [Online]. Available: http://www.ons.org.br/AcervoDigitalDocumentosEPublicacoes/Resultados_PPR_2019_.pdf.

[2] ONS, Comunicação Verbal na Operação. 2020. Accessed on: Mar 07, 2021. [Online]. Available: http://www.ons.org.br/%2FMPO%2FDocumento%20Normativo%2F4.%20Rotinas%20Operacionais%20-%20SM%205.13%2F4.1.%20Rotinas%20Gerais%2F4.1.7.%20Relacionamento%20Operacional%2FRO-RO.BR.01_Rev.12.pdf.

[3] F. Filippidou and L. Moussiades, "A Benchmarking of IBM, Google and Wit Automatic Speech Recognition Systems.," in AIAI (1), 2020, vol. 583, pp. 73–82.

[4] IBM Watson, Speech to Text, 2015. Accessed on: Mar 07, 2021. [Online]. Available: https://www.ibm.com/cloud/watson-speech-to-text.

[5] Google, Speech to Text, 2017. Accessed on: Mar 07, 2021. [Online]. Available: https://cloud.google.com/speech-to-text.

[6] Wit, Build Natural Language Experiences, 2013. Accessed on: Mar 07, 2021. [Online]. Available: https://wit.ai.

[7] T. Viarbitskaya and A. Dobrucki, "Audio processing with using Python language science libraries", 2018 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), Poznan, Poland, 2018, pp. 350-354, doi: 10.23919/SPA.2018.8563430.

[8] A. C. Morris, V. Maier, and P. D. Green, "From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition.," in INTERSPEECH, 2004.

[9] S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. Beijing: O'Reilly, 2009.

[10] D. Rao, B. McMahan, Natural Language Processing with PyTorch, O'Reilly Media, 2019.

[11] Pedregosa et al., Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830, 2011.

[12] C. Howard, H. Lane, H. Hapke, Natural Language Processing in Action: Understanding, analyzing, and generating text with Python, Manning Publications, 2019.

[13] J. D. Rennie, L. Shih, J. Teevan, and D. R. Karger, "Tackling the Poor Assumptions of Naive Bayes Text Classifiers.," in ICML, 2003, pp. 616–623.

[14] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A Library for Large Linear Classification," Journal of Machine Learning Research, vol. 9, 2008.

[15] Y. Tsuruoka, J. Tsujii, and S. Ananiadou, "Stochastic gradient descent training for L1-regularized log-linear models with cumulative penalty," in Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - ACL-IJCNLP '09, 2009.

[16] K. Taunk, S. De, S. Verma, and A. Swetapadma, "A brief review of nearest neighbor algorithm for learning and classification," in 2019 International Conference on Intelligent Computing and Control Systems (ICCS), 2019, pp. 1255–1260.

[17] F. Murtagh, "Multilayer perceptrons for classification and regression," Neurocomputing, vol. 2, no. 5–6, pp. 183–197, 1991.

[18] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[19] H. Zhang, "The Optimality of Naive Bayes," in Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2004), 2004.

[20] C.-C. Chang and C.-J. Lin, LIBSVM: a library for support vector machines. 2001.

[21] G. James, D. Witten, T. Hastie, and R. Tibshirani, An Introduction to Statistical Learning -- with Applications in R, vol. 103. New York: Springer, 2013.

[22] H. Robbins and S. Monro, "A Stochastic Approximation Method," The Annals of Mathematical Statistics, vol. 22, no. 3, pp. 400--407, 1951.

[23] L. Bottou and O. Bousquet, "Learning using Large Datasets.," in NATO ASI Mining Massive Data Sets for Security, 2007, vol. 19, pp. 15–26.

[24] E. A. Wan, "Discrete time neural networks," Appl. Intell., vol. 3, no. 1, pp. 91–105, 1993.

[25] T. G. Dietterich, "Ensemble Methods in Machine Learning," in Multiple Classifier Systems, Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 1–15.

[26] R. R. Bouckaert and E. Frank, "Evaluating the Replicability of Significance Tests for Comparing Learning Algorithms.," in *PAKDD*, 2004, vol. 3056, pp. 3–12.

[27] B. H. Shekar and G. Dagnew, "Grid search-based hyperparameter tuning and classification of microarray cancer data," in 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP), 2019, pp. 1–8.

[28] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, F-score and ROC: A family of discriminant measures for performance evaluation," in Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1015–1021.

[29] T. R. Shultz et al., "Confusion Matrix," in Encyclopedia of Machine Learning, Boston, MA: Springer US, 2011, pp. 209–209.