Comparison of Supervised Machine Learning Techniques for CERN CMS Offline Data Certification

Mantas Stankevicius, Virginijus Marcinkevicius, and Valdas Rapsevicius

Vilnius University, Faculty of Mathematics and Informatics, Lithuania,
mantas.stankevicius@cern.ch

Abstract. The Compact Muon Solenoid (CMS) is one of the experiments at the CERN Large Hadron Collider (LHC). CMS is a general-purpose detector able to record particle collisions up to 40 million times each second. 40 MHz rate results in approx 40 Tb/s, however journey of recorded CMS data is long and only small fraction makes to the final step - physics analysis. Chain of automated and semi-automated processes filter, reconstruct, calibrate and verify data before it can be used for physics analysis. Data certification process starts with data aggregation in histograms, plots and various statistical quantities, and finishes with manual data quality assessment and decision. During the last step multiple experts review and evaluate (certify) data as being "good" or "bad". This results in high manpower demand and occasional involuntary human errors. Data certification process consists of online (to determine possible problems during the data taking, only the fraction of data is used) and offline (not constrained in time full data set analysis including physics data reconstruction). Main goal of this research is to investigate the applicability and provide the comparison of various supervised machine learning techniques for offline data certification process automation. Removal or significant reduction of manual labor and exclusion of human errors from the CMS data certification process are the main motivations of this research.

Keywords: CERN CMS, Data certification, Supervised machine learning, Neural network.

1 Problem Statement

Data Quality Monitoring (DQM) is one of the central and the critically important projects of the Compact Muon Solenoid (CMS) detector at the CERN Large Hadron Collider (LHC). Main goal of the DQM is to provide and support a single end-to-end process for reliable certification of the recorded data. The certification process comprises of multiple parts, starting from filling and handling multiple histograms and scalar monitor elements and finishing with the list of "good" list of runs and lumisections. The goal of the DQM is to discover

Lupeikiene A., Matulevicius R., Vasilecas O. (eds.):
Baltic DB&IS 2018 Joint Proceedings of the Conference Forum and Doctoral Consortium.

and pinpoint errors, problems occurring in detector hardware or reconstruction software, early, with sufficient accuracy and clarity to maintain good detector and operation efficiency [10].

While online DQM process only the small fraction of sampled data for immediate response and provides just technical Detector Performance Group (DPG) histograms, the offline DQM examines the full dataset and generates both DPG and physics POG (Physics Objects Group) histograms. Output is being accumulated into the ROOT [4] format file and displayed to shifters and experts by using DQMGUI application [12]. Then the certification process involves shifters for monitoring various sets of histograms in online and offline and marking collected data as "good" or "bad" and providing comments. Certification is made on run (data taking session) and luminosity section (lumisection) levels. Lumisection is a data sample of approximately 23 seconds of data-taking. Final decision is made by the data certification group experts by cross-checking assessments and generating the final list of good data [3] also known as GoldenJSON. Data quality flags, comments and other related information is stored in Run Registry database [11].

Since the start of the experiment in 2008 the CMS DQM processes and tools have been worked out and are stable. A decade of data taking have accumulated the sufficient amount of labeled DQM data. This allows the scientists to examine the possibility to automate the final step of the data certification process by using the recent advances in Machine Learning algorithms for binary classification. In the current system, hundreds of different histograms come as an input and the output is one of “good” or “bad” for each lumisection.

2 Related Work

Currently, there are several initiatives trying different approaches for the data certification and anomaly detection. One of the most advanced research is done by Yandex School of Data Analysis [2]. They investigate applicability of deep neural networks for anomaly detection and classification. Predictive power of neural network is quite high - ROC AUC score equals to 0.96. The research is based on data collected in 2010 by CERN CMS experiment. In this paper we use newer and bigger dataset and compare other supervised machine learning algorithms.

3 Methodology

Data used in this research was collected by CERN CMS [7] experiment at LHC during 2016 data-taking. CMS DQM group provides aggregated data of JetHT2016 dataset and so-called GoldenJSON. JetHT2016 dataset is widely used at CMS for machine learning research. Dataset contains around 160,000 lumisections. GoldenJSON is a JSON file which contains labels for lumisections. Labels are “good” or “bad” meaning if lumisection is good for further physics analysis or not. Each lumisection consists of 401 parameters (histograms) which are vectors of 7 numbers (mean, RMS and 5 quantiles of the histogram). So in total each lumisection has 2807 features. There are several other parameters used for data preprocessing but not for training, for example run number and lumisection number, etc.

Data preprocessing consists of two steps: merging dataset with GoldenJSON and feature normalization. We used standard score normalization to center and scale data so that distribution of each feature has a mean value 0 and standard deviation of 1.

3.1 Class Imbalance Problem

Dataset contains very small amount (1.8%) of so-called "bad" lumisections, hence class distribution ratio is 49:1. Therefore, multiple tactics are used to deal class imbalance: class weights penalty, stratified fold cross validation and rank based performance metrics (see Section 3.3).

First of all, we introduced class weights during training of a model. Additional penalty for classification mistakes can affect model to pay more attention to minority class.

Secondly, we use stratified cross validation to preserve percentage of samples for each class during cross validation. Natural distribution of classes in train and test splits helps model to fight class imbalance during training. However, classical stratified cross validation is limited to a certain number of splits. We decided to use StratifiedShuffleSplit [5] as it gives randomized folds. Using the randomized folds model can be validated multiple times while keeping same train-test split ratio.

3.2 Classification Methods

Five methods were selected for a comparison: Support Vector Machine, Random Forest, Naive Bayes, Gradient Boosted Trees and Artificial Neural Network. We chose popular methods from these categories: probabilistic, ensemble and hierarchical.

Support Vector Machine (SVM) We use scikit-learn [5] implementation of Support Vector classifier. However after series of tries SVM turned out to be inappropriate for the task. Large number of high-dimensional training data badly affected performance, therefore this method was eliminated from further development and evaluation.

Random Forest (RF) We use scikit-learn implementation of Random Forest classifier. The forest has 64 trees with depth of 7 layers, these are hand-picked parameters.

Gradient Boosted trees (XGB) We use XGBoost [8] implementation for gradient boosted decision trees. Same as with Random Forest, we hand-picked several parameters to improve performance: number of trees is 64 with max depth of 7 layers.

Gaussian Naive Bayes (NB) We use scikit-learn implementation of Gaussian Naive Bayes algorithm for classification. The method trains very fast, but its predictive power is too poor for this task.

Artificial Neural Network (ANN) We use Keras [9] library and Tensorflow [1] backend. Neural network consists of 3 hidden layers. Each regular densely-connected NN layer uses rectified linear unit (ReLU) activation function and is followed by Dropout layer. Output layer uses sigmoid activation function (see Fig. 1). Early stopping is used to avoid over-fitting and stop training when validation accuracy begins to decrease.

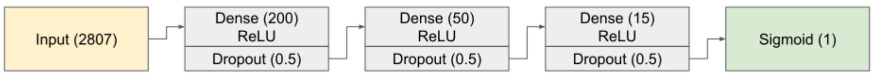

Fig. 1. Neural network schema

3.3 Performance Metrics

To determine the performance of different methods we use three performance measures: accuracy (ACC), F_1 score and Receiver Operating Characteristic (ROC) with Area Under Curve (AUC). For curiosity only, we track training time as well.

Threshold based. Accuracy (ACC) is a very simple and an intuitive method as it measures actual predicted values. However, that makes ACC poor metric for imbalanced data.

F_1 score is a weighted average of precision and recall. Both false positives and false negatives are taken into account which makes F_1 score usually more useful than accuracy.

Rank based. Receiver Operating Characteristic (ROC) with Area Under Curve (AUC) measures how well positive cases are ordered before negative cases [6]. That makes it a great metric to evaluate model performance having uneven class distribution.

4 Experimental Results

Experimental Setup. Software used: Python (v3.6), Keras (v2.1.5), Tensorflow (v1.7), XGBoost (v0.71), scikit-learn (v0.19.1). Hardware used: PC with NVIDIA GPU (GeForce GTX 1080 Ti) and virtual machine (8 cores 2.2 GHz, 16 GB RAM)

Primary task of this research is to compare supervised machine learning models and find the most applicable for data certification. Each model was cross-validated for 10 times. Support Vector Classifier was eliminated at early stage due to excessive training time. Naive Bayes classifier was eliminated second, because of poor performance for this task. Other 3 models show good performance with ROC AUC score over 0.95. The best model which works almost out-of-the-box is Gradient Boosted Trees (XGB) with average ROC AUC score 0.987. Average ROC AUC scores are shown in Fig. 2. Full details about each model and each metric is in Table 1. Best scores are in bold.

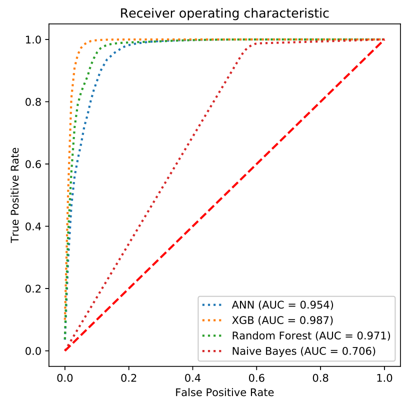

Fig. 2. Average ROC AUC using shuffle stratified 10 fold cross validation

Table 1. Performance metrics. Mean scores and standard deviations (±)

	AUC	±	ACC	±	F_1	±	time	±
XGB	**0.987**	0.004	**0.997**	0.000	**0.998**	0.000	108.09	2.621
Random Forest	0.970	0.004	0.980	0.001	0.990	0.000	**44.925**	2.490
ANN	0.954	0.005	0.961	0.015	0.979	0.008	130.236	38.413
Naive Bayes	0.706	0.008	0.971	0.002	0.985	0.001	10.529	1.289

Secondary task follows the hypothetical automatic classification process in a way that model is trained on historical data and only then it is used to classify the new incoming data. In order to mimic this use-case we sorted dataset timewise and split into 80:20. All models were trained using same split. Fig. 3 shows ROC curves with AUC value. All models perform quite well with 94+% AUC score.

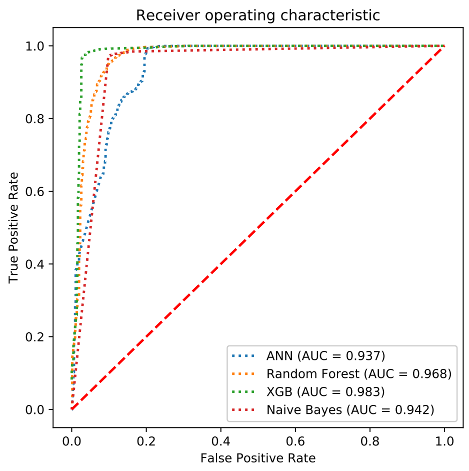

Fig. 3. ROC curves with AUC of sorted dataset and split of 80:20

However due to lucky train-test set distribution Naive Bayes model performs so well, nearly 20% better than average (see Fig. 2). This behavior once again proves necessity of cross validation for a proper model validation.

5 Conclusions and Future Works

In this paper we ran experiments with 5 classification methods and compared their performance on CERN CMS JetHT2016 dataset. Best performing model is Gradient Boosted Trees which ROC AUC score equals 0.987. It seems to be a sufficient score already, but due to imbalanced class distribution (98.2% to 1.8%) it is barely better than "most popular class" classifier. Since only manual search of hyper-parameters was performed, therefore we believe that full potential of deep neural network is not yet discovered. Further grid-search of hyper-parameters (learning rate, dropout rate, batch size, number of neurons, hidden layers, epochs, etc) should be done to improve performance and predictive power.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), https://www.tensorflow.org/, software available from tensorflow.org
2. Azzolini, V., et al.: Deep learning for inferring cause of data anomalies (2017), http://inspirehep.net/record/1637193/files/arXiv:1711.07051.pdf
3. Borrello, L.: The Data Quality Monitoring Software for the CMS experiment at the LHC. Tech. Rep. CMS-CR-2014-431, CERN, Geneva (Nov 2014), http://cds.cern.ch/record/2121269
4. Brun, R., Rademakers, F.: ROOT: An object oriented data analysis framework. Nucl. Instrum. Meth. **A389**, 81–86 (1997). https://doi.org/10.1016/S0168-9002(97)00048-X
5. Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., Varoquaux, G.: API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning. pp. 108–122 (2013)
6. Caruana, R., Niculescu-Mizil, A.: An empirical comparison of supervised learning algorithms. In: Proceedings of the 23rd International Conference on Machine Learning. pp. 161–168. ICML '06, ACM, New York, NY, USA (2006). https://doi.org/10.1145/1143844.1143865, http://doi.acm.org/10.1145/1143844.1143865
7. Chatrchyan, S., et al.: The CMS Experiment at the CERN LHC. JINST **3**, S08004 (2008). https://doi.org/10.1088/1748-0221/3/08/S08004
8. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 785–794. KDD '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939785, http://doi.acm.org/10.1145/2939672.2939785
9. Chollet, F., et al.: Keras. https://keras.io (2015)
10. De Guio, F.: The CMS Data Quality Monitoring software experience and future improvements. Tech. Rep. CMS-CR-2013-350, CERN, Geneva (Oct 2013), http://cds.cern.ch/record/2194526
11. Rapsevicius, V.: CMS Run Registry: Data Certification Bookkeeping and Publication System. Tech. Rep. CMS-CR-2011-020, CERN, Geneva (Jan 2011), http://cds.cern.ch/record/1345306
12. Rovere, M.: The Data Quality Monitoring Software for the CMS experiment at the LHC. J. Phys.: Conf. Ser. **664**(7), 072039. 8 p (2015), http://cds.cern.ch/record/2159196