Proceedings of the Federated Conference on Computer Science and Information Systems pp. 245–248
DOI: 10.15439/2016F152
ACSIS, Vol. 8. ISSN 2300-5963

Automatic Feature Engineering for Prediction of Dangerous Seismic Activities in Coal Mines

Eftim Zdravevski, Petre Lameski, Andrea Kulakov
Faculty of Computer Science and Engineering
Ss.Cyril and Methodius University, Skopje, Macedonia
Email: {eftim.zdravevski,petre.lameski,andrea.kulakov}@finki.ukim.mk

***Abstract*—In this paper we present our submission to the AAIA'16 Data Mining Challenge, where the objective was to predict dangerous seismic events based on hourly aggregated readings from different sensor and recent mining expert assessment of the conditions in the mine. During the course of the competition we have exploited a framework for automatic feature extraction from time series data that did not require any manual tuning. Furthermore, we have analyzed the impact of overlapping of input data on model robustness. We argue that training an ensemble of classifiers with distinct (i.e. non-overlapping) chronological data rather than one classifier with all available data can produce more reliable and robust prediction models. By doing that, we were able to avoid overfitting and obtain the same score performance on the evaluation and test datasets, despite the significant data drift in the datasets.**

***Keywords*—feature engineering, feature selection, time series classification, temporal data mining, drift detection**

I. Introduction

The task in the AAIA'16 Data Mining Competition [1] is to devise a reliable prediction model for detecting periods of increased seismic activity that endangers miners working underground in coal mines. The data set consists of hourly aggregated readings from seismic sensors that count the number of seismic bumps perceived at longwalls and measure their total energy. Data records are composed of 24 consecutive hours of such readings coupled with the most recent assessments of the conditions at the longwalls made by mining experts. The target attribute in the data corresponds to information whether in a following period of 8 hours the total energy of seismic bumps exceeds a warning level. The full training dataset contains 133151 records with 541 columns, each corresponding to 24 hours of measurements. The evaluation metric of the competition was Area Under the ROC Curve (AUC).

The challenges of the competition were versatile: quite different distribution of working sites in the training and tests dataset that indicated of potential distribution drift in the data; the training class distribution was imbalanced; the final training set was 40 times larger than the test set; and the class distribution in the test set was unknown. After the competition ended it was disclosed that 2% of the records in the training and 5% in the test set were warnings. These challenges required very robust features and prediction models that would prevent over-fitting to the training set. In this paper we describe how our submission to the challenge addressed these challenges.

II. Cross-validation for model selection

In order to evaluate the models and feature sets locally, a way that will take into consideration the distribution of working sites, classes and train/test dataset size is required. To address this challenge, we developed a strategy that splits the training set keeping in mind the following parameters: general class distribution in the training set; class distribution per working site in the training set; and ratio of unknown versus known working sites in the test set.

Using these parameters, we tried to replicate a split of the train set into two sets, one for training and one for local testing, that would resemble the relation between the real training and test sets. When splitting the training set, we iteratively choose whether to add the working site to the train or test split based on its working class distribution, the current and desired class distribution in the splits, and the known vs unknown ratio in the test split. Using this approach, we were able to get a more realistic estimate of the feature set performance than the cross-validation schemes. We considered cross-validation scores to be more realistic if they resembled the leaderboard score, and thus were lower than the ones obtained with regular cross-validation. However, even with different feature subsets it was always 6% greater than the leaderboard score (always over 0.98).

Even though it was promising approach, we did not have time to develop a stratified or leave-one-subject-out cross-validation strategy based on it, so we have eventually abandoned it and relied for performance estimation based on the leaderboard score. In addition, this strategy based on the assumption that the training and test sets have similar class distribution, which was not specified in the task description. After the competition ended and the test labels were disclosed, it turned out that this assumption did not hold indeed.

III. Feature Engineering Framework

This competition coincided with our work on a framework for automated feature extraction from time series data. Therefore, it presented a good opportunity to test an early prototype of the framework on this dataset.

A. Feature generators

Using a systematic approach, the system is able to generate a variety of features that can robustly describe the dataset. A recent data mining competition for posture recognition of firefighters [2] inspired different feature engineering approaches

that are very effective [3, 4, 5]. Using the proposed approaches there, from each series of readings the system generates the following types of features: basic statistics (minimum, maximum, range, arithmetic mean, harmonic mean, geometric mean, median, mode, standard deviation, variance, skewness, kurtosis, signal-to-noise ratio, energy, etc.); curve fitting parameters [4]; equal-width histogram features [5]; percentile based features (first quartile, median, third quartile, inter-quartile range, amplitude, etc.) [3]; auto-correlation of the signal with several types of correlations (signal processing auto-correlation and Pearson, Spearman and Kendall correlation) [4]; and inter-correlations between each pair of raw time series values using the aforementioned types of correlation coefficients.

B. Time series generators

The feature extraction framework is able to generate new time series and then based on them to extract new sets of features, just like from an original time series. Authors of [3, 4, 5] demonstrated that this approach can further enhance the predictive performance of the system. Therefore, the framework uses the following time series generators (TSG): first derivatives of the original series [5]; amplitudes, frequencies and magnitudes obtained with fast Fourier transformation [3]; delta series (the deviations of the original values from the mean of the particular block of the time series), which can remove the seasonality in the data; sliding window time series [6]; and combining two time series by multiplying, subtracting or dividing their values [4].

C. Learning algorithms

For estimating the informativeness of individual features and the predictiveness of the whole problem the framework uses Random Forest (RF) [7] and Extremely Randomized Trees (ERT) [8] with high number of trees. They are used with the default parameters, as we have noticed that tuning them does not improve the performance dramatically (unlike with SVMs, for example). ERT models are very similar to RF in terms of predictive performance, but quite faster. We have noticed that when using the same number of trees ERT is significantly faster (over 50%) than RF, especially when the number of features is large (over 500).

D. Feature extraction and selection heuristics

Applying all possible feature transformations would generate very large number of features and would make learning based on them practically impossible. To mitigate this, the feature engineering and selection processes are interleaved, so generation of new time series and features is heuristically guided.

The algorithm for feature extraction and selection is shown in Figure 1. After the initialization and configuration of which features should be computed, the processing starts, one dataset instance (record) at a time. When the features are extracted from the whole dataset, it estimates the predictive performance and calculates feature importances in order to prepare a baseline for the feature selection loop that follows.

To improve the performance of the model under data drift, the framework performs greedy wrapper feature selection, inspired by the idea proposed in [9]. First it merges the training and validation or test dataset and generates an artificial "drift" class, denoting whether the instance is from the training or the test dataset. Then it estimates the importance of each feature when predicting the regular class (normal/warning) and the drift class (train/test). Afterwards, it calculates a set of feature importance thresholds for the regular class and the drift class. Next, in a loop it evaluates the performance different sets of features based on whether they fall within these importance thresholds.

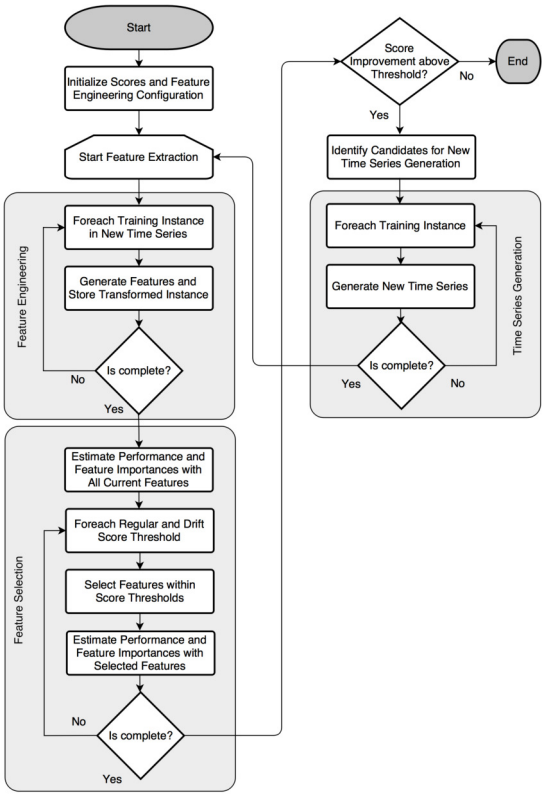

Fig. 1. Algorithm for feature extraction and selection

Next, by averaging the regular informativeness of the retained features for a particular time series the framework estimates which time series are informative for prediction of the target class. Additionally, it calculates correlations between different time series. Based on these heuristics, it identifies candidates for new which new series should be generated. After the new series are generated, the algorithm returns to the feature engineering phase. The second time it only generates features with only those transformations that resulted in retained features in the initial run. Afterwards, it merges the new feature set with the selected features in the previous iteration. The feature engineering and selection loop ends when generating new features does not improve the best performance by a considerable margin. For this competition, the we have only used the initial and one additional loop of generation of features.

Given that we were not able to replicate the training/test split for performing local evaluations, we decided to use the leaderboard scores for selecting the feature importance thresholds. By default, the framework uses cross-validation.

IV. The Feature Engineering Framework in Practice

The framework implementation it is still not publicly available. It has been implemented in Python on top of the scikit-learn Library [10]. In order to use the framework, we only need to specify the data set information: number and indexes of nominal features and numeric features, the number of time series and samples per series, the CSV files containing the description of columns and series. Then the framework takes over. It starts by calculating the basic statistics, inter-correlations, autocorrelations, histograms and quantiles. Based on them it evaluates which time series are informative. In this case, it discovered that the time series can be ranked as shown in Table I (columns IRI and IR - initial relative importance and initial rank, respectively).

This drift detection mechanism turned out to be very helpful for feature reduction. Namely, depending on the particular feature set size, removing the features that are good drift predictors improved our leaderboard score by 1-3 %. In our final solution the threshold 30 for both scores turned out to give good results thus removing almost 60% of the generated features. After their removal, the relative importance of different time series changed dramatically compared to the initial ranking, as can seen in Table I (columns IR and FR - initial and final rank, respectively).

A. Importance of time series

Some of the time series were significantly less informative than others, so they were discarded, keeping only the top 16 series. When evaluating the performance locally without the leaderboard, the system was able very quickly to find a feature set with AUC ROC score over 0.99 with stratified and regular cross-validation. When submitting those predictions to the public leaderboard we were able to get to a score of about 0.91. This dramatic drop of performance was a clear evidence of drift in the training and test datasets, which was somewhat expected due to the different mining sites.

TABLE I. Time series importance before and after importance evaluations. IRI=Initial Relative Importance, IR=Initial Rank, FRI=Final Relative Importance, FR=Final Rank

Time Series Name	IRI	IR	FRI	FR
max_gactivity	1.000	1	1.000	1
avg_difference_in_gactivity	0.998	2	0.788	9
max_difference_in_gactivity	0.962	3	0.786	10
avg_difference_in_genergy	0.954	4	0.771	11
max_difference_in_genergy	0.938	5	0.766	12
avg_genergy	0.916	6	0.886	4
max_genergy	0.889	7	0.902	3
avg_gactivity	0.885	8	0.909	2
highest_bump_energy	0.663	9	0.861	6
sum_e2	0.410	10	0.742	15
sum_e3	0.345	11	0.874	5
total_number_of_bumps	0.310	12	0.793	8
sum_e4	0.192	13	0.766	13
count_e2	0.175	14	0.646	16
count_e3	0.140	15	0.744	14
count_e4	0.116	16	0.811	7
sum_e5	0.040	17		
count_e5	0.022	18		
sum_e6plus	0.017	19		
count_e6plus	0.013	20		
number_of_distressing_blasts	0.004	21		
number_of_rock_bursts	0.000	22		

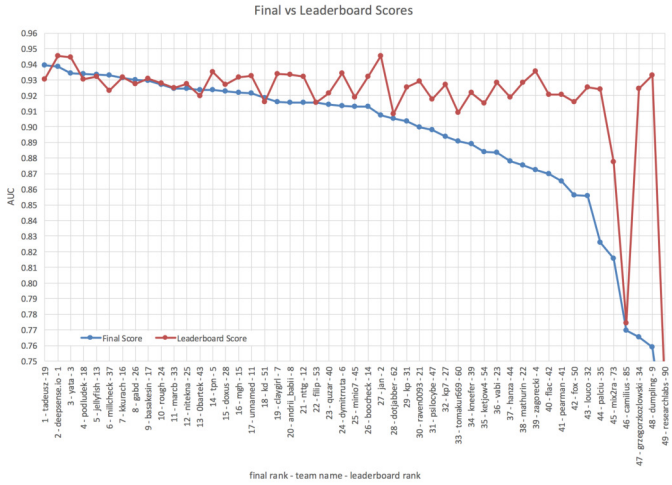

Fig. 2. Final vs. Leaderboard scores of the competitors.

As described earlier, the cross-validation scores were significantly different the leaderboard scores. On the other hand, different feature subsets performed similarly to each other with cross-validation and on the leaderboard. Because of those two reasons and the limited time we had for spending on this competition we decided to use only the features of the original time series and disable generation of new time series.

B. Oversampling in datasets

The fact that the size of the training set was significantly larger than the test set, pointed out that there must be some over-sampling in the training instances. To investigate this, we have analyzed the training dataset comparing consecutive rows that are for the same working site. We have discovered that in most cases the consecutive rows are shifted by 1 hour, thus overlapping for 23 hours. However, the test instances do not have any overlapping (or at least it can not be discovered). This mean that the training and test dataset are not identically and independently distributed. That combined with the over-sampling posed significant risk of over-fitting. In order to alleviate that, we have split the training set into 24 folds of instances that do not overlap. In particular, if we have consecutive overlapping instances (1 shift, 23 hours overlap) numbered with $1..n$ from the same working site, in the i-th fold the following instances would be assigned: $i, i+s, i+2\times s, ...$, where $s = 24$ is the number of folds. Likewise, we have tested with 12, 8, 6, 4 and 2 folds. Then, having s folds we train a separate classifier on each fold and then average the predictions of all s classifiers. Regardless of the feature subset and used classifier, when using 24 folds gave best results, about 1-3% more than training one classifier on the complete training set (i.e. only one fold). This was very peculiar discovery that should be considered when creating batches from continuous streams of data. A similar effect of over-sampling in datasets was discovered and explained in [11].

C. Final submission

The predictions based on different feature subsets and different training data subsets scored similarly on the public leaderboard. In order to diversify the predictions and aiming

to achieve more robust prediction models, we aggregated the results from 5 different predictions: 4 ERT models trained with various feature subsets that had 200-900 features (various basic statistics, histograms, percentiles, linear and quadratic fit coefficients, correlations, etc. obtained from the 16 retained time series listed in Table I) and 1 logistic regression model trained with only 21 features (reduced from the larger feature set with Correlation-based Feature Selection). This improved our leaderboard score to 0.9276, about 1.5% better than any the individual classifiers that were used in the ensemble. Some classifiers (e.g. logistic regression) made predictions close to 0 while all others were predicting exactly 0 (the probability of warning). To leverage these properties, we decided to determine whether the prediction is 0 or not with simple majority vote. If not then the individual predictions are averaged. Instead of averaging the predictions in such cases, predicting 0 improved the leaderboard score by 0.5%.

As it can be seen from Figure 2, many of the teams had dramatically over-fitted their models to the training and leaderboard datasets, thus their performance of the final dataset dramatically dropped. Only few teams were able to produce the same or better score on the final dataset as on the leaderboard.

V. Conclusion

This competition provided an interesting and interactive opportunity to tackle various challenges encountered in real-world data analysis. Our initial goal was to test if automatic feature extraction and selection from time series data would be feasible for such problem. We were able to discover and understand interesting patterns in the data, such as the ranking of importance of time series or other important features like expert seismic assessments. Additionally, we discovered the importance of overlapping in the training dataset in relation to over-fitting. Another interesting realization was that even simple classifiers like logistic regression with very simple features can give leaderboard score of about 0.92. Our best leaderboard score was average of predictions of several different classifiers and/or different feature sets and it was about 0.928.

It is important to point out that the process of automatically generating features did not require manual tuning. Furthermore, the framework was able to generate and recognize informative features and time series, as well as to retain only features that are robust to drift in the data. The only manual work that we performed was related to the cross-validation experiments and for submitting solutions to the competition system. We consider that the obtained performance is significant due to the automatically generated features. We acknowledge that the framework still requires work to be done in relation to more efficient searching of redundant features and more optimal iterative generation of new features. Our experiments with other datasets confirms that it is able to match the best published performance, or in some cases to even improve them.

References

[1] A. Janusz, M. Sikora, Ł. Wróbel, and D. Ślezak, "Predicting Dangerous Seismic Events: AAIA16 Data Mining Challenge," in *Proceedings of FedCSIS 2016*. IEEE, 2016, in print September 2016.

[2] M. Meina, A. Janusz, K. Rykaczewski, D. Slezak, B. Celmer, and A. Krasuski, "Tagging firefighter activities at the emergency scene: Summary of aaia'15 data mining competition at knowledge pit," in *Computer Science and Information Systems (FedCSIS), 2015 Federated Conference on*, Sept 2015. doi: 10.15439/2015F426 pp. 367–373.

[3] J. Lasek and M. Gagolewski, "The winning solution to the aaia'15 data mining competition: Tagging firefighter activities at a fire scene," in *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems*, ser. Annals of Computer Science and Information Systems, M. P. M. Ganzha, L. Maciaszek, Ed., vol. 5. IEEE, 2015. doi: 10.15439/2015F418 pp. 375–380.

[4] A. Zagorecki, "A versatile approach to classification of multivariate time series data," in *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems*, ser. Annals of Computer Science and Information Systems, M. P. M. Ganzha, L. Maciaszek, Ed., vol. 5. IEEE, 2015. doi: 10.15439/2015F419 pp. 407–410.

[5] E. Zdravevski, P. Lameski, R. Mingov, A. Kulakov, and D. Gjorgjevikj, "Robust histogram-based feature engineering of time series data," in *Computer Science and Information Systems (FedCSIS), 2015 Federated Conference on*, ser. Annals of Computer Science and Information Systems, M. P. M. Ganzha, L. Maciaszek, Ed., vol. 5. IEEE, Sept 2015. doi: 10.15439/2015F420 pp. 381–388.

[6] M. Grzegorowski and S. Stawicki, "Window-based feature engineering for prediction of methane threats in coal mines," in *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, ser. Lecture Notes in Computer Science, Y. Yao, Q. Hu, H. Yu, and J. W. Grzymala-Busse, Eds. Springer International Publishing, 2015, vol. 9437, pp. 452–463. ISBN 978-3-319-25782-2

[7] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[8] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, 2006. doi: 10.1007/s10994-006-6226-1

[9] M. Boullé, "Tagging fireworkers activities from body sensors under distribution drift," in *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems*, ser. Annals of Computer Science and Information Systems, M. Ganzha, L. Maciaszek, and M. Paprzycki, Eds., vol. 5. IEEE, 2015. doi: 10.15439/2015F423 pp. 389–396.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[11] M. Boullé, *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing: 15th International Conference, RSFDGrC 2015, Tianjin, China, November 20-23, 2015, Proceedings*. Cham: Springer International Publishing, 2015, ch. Prediction of Methane Outbreak in Coal Mines from Historical Sensor Data under Distribution Drift, pp. 439–451. ISBN 978-3-319-25783-9