DDBJ Data Analysis Challenge: a machine learning competition to predict *Arabidopsis* chromatin feature annotations from DNA sequences

Eli Kaminuma*, Yukino Baba, Masahiro Mochizuki, Hirotaka Matsumoto, Haruka Ozaki, Toshitsugu Okayama, Takuya Kato, Shinya Oki, Takatomo Fujisawa, Yasuazuka Nakamura, Masanori Arita, Osamu Ogasawara, Hisashi Kashima and Toshihisa Takagi

1Center for Information Biology, National Institute of Genetics, 1111 Yata, Mishima, Shizuoka 411-8540, Japan
2Graduate School of Informatics, Kyoto University, 36-1 Yoshida-Honmachi, Sakyo, Kyoto 606-8501, Japan
3IMSBIO Co., Ltd., 4-21-1, Higashi-ikebukuro, Toshima, Tokyo 170-0013, Japan
4Laboratory for Bioinformatics Research, RIKEN Center for Biosystems Dynamics Research, 2-1 Hirosawa, Wako, Saitama 351-0198, Japan
5BITS Co., Ltd., 1-5-5 Surugadai, Kanda, Chiyoda, Tokyo 101-0062, Japan
6Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo, Tokyo 113-8656, Japan
7Graduate School of Medical Sciences, Kyushu University, 3-1-1 Maidashi, Higashi, Fukuoka 812-8582, Japan
8Graduate School of Science, The University of Tokyo, 7-3-1 Hongo, Bunkyo, Tokyo 113-0033, Japan

(Received 22 July 2019, accepted 1 November 2019; J-STAGE Advance published date: 26 March 2020)

Recently, the prospect of applying machine learning tools for automating the process of annotation analysis of large-scale sequences from next-generation sequencers has raised the interest of researchers. However, finding research collaborators with knowledge of machine learning techniques is difficult for many experimental life scientists. One solution to this problem is to utilise the power of crowdsourcing. In this report, we describe how we investigated the potential of crowdsourced modelling for a life science task by conducting a machine learning competition, the DNA Data Bank of Japan (DDBJ) Data Analysis Challenge. In the challenge, participants predicted chromatin feature annotations from DNA sequences with competing models. The challenge engaged 38 participants, with a cumulative total of 360 model submissions. The performance of the top model resulted in a score of 0.95. Over the course of the competition, the overall performance of the submitted models improved by an AUC score of 0.30 from the first submitted model. Furthermore, the 1st- and 2nd-ranking models utilised external data such as genomic location and gene annotation information with specific domain knowledge. The effect of incorporating this domain knowledge led to improvements of approximately 5%–9%, as measured by the AUC scores. This report suggests that machine learning competitions will lead to the development of highly accurate machine learning models for use by experimental scientists unfamiliar with the complexities of data science.

**Key words:** chromatin features prediction, deep learning, machine learning competition, sequence read archive

**INTRODUCTION**

The next-generation sequencers that appeared in 2005 have rapidly been developed to reduce the costs of analyses and realise the goal of the 1000 dollars human genome (DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program [http://www.genome.gov/sequencingcostsdata]) (van Nimwegen et al., 2016). As these large-scale sequences and sequence annotations are accumulated, experimental researchers increasingly
seek to construct machine learning models. However, many experimental researchers are not familiar with the analytical techniques of data science, and they often lack the connections to find data scientists with whom to collaborate. One solution to this problem is the crowdsourcing approach, i.e., outsourcing these tasks as challenges to the crowd (Mayer et al., 2011; Saez-Rodriguez et al., 2016; Wazny, 2017). By holding contest-style crowdsourcing events in the form of ‘machine learning competitions’, experimental researchers can leverage the power of crowd workers (Bender, 2016). In the domain of biomedical research, many crowdsourcing competition events have been held, such as the CASP project for protein structure prediction (Moult, 2005), the BioCreAtIvE project for text mining methods in molecular biology (Hirschman et al., 2005), the CAPRI project for protein-protein docking (Janin, 2005) and the DREAM project for general tasks in systems biology and medicine (Prill et al., 2011). There are also crowdsourcing platform services specifically for data science. These include Kaggle (https://www.kaggle.com), which conducts machine learning competitions such as the biomedical task ‘American Epilepsy Society Seizure Prediction Challenge’, which attracted 504 teams with 654 competitors (Brinkmann et al., 2016). However, one issue with these crowdsourcing competition projects is that they typically only report their results on websites and are only occasionally documented in academic journals with an information science background. While guidelines for running computational competitions (Friedberg et al., 2015) already exist, they provide only an outline of competition rules. Thus, it is difficult for experimental scientists to obtain concrete procedures as competition protocols and to develop further knowledge regarding the organisation of competitions and participants. Meanwhile, there is an education-based data science platform, the UniversityOfBigData (Baba et al., 2018), which is open to the public and features leaderboard panels with participant nicknames, model performance scores and submission status. The platform was constructed for educational purposes and participants are able to visually monitor submission activity and improvements in model performance.

In this report, we conducted a machine learning competition, the DDBJ Data Analysis Challenge, to investigate the quantitative evaluation of participant contributions and the improvement of submitted models for the purpose of supporting experimental scientists unfamiliar with data science. The DDBJ (Kodama et al., 2018) maintains large-scale DNA sequence databases such as the Sequence Read Archive (SRA) (Kodama et al., 2012) for members of the International Nucleotide Sequence Database Collaboration (Cochrane et al., 2016). The machine learning competition was conducted using the UniversityOfBigData platform and took place during the summer of 2016. The task of the challenge was to predict chromatin feature annotations from DNA sequences while referring to the data protocol of a secondary annotation database of DDBJ sequences. We can obtain chromatin feature annotations as peak-call regions based on ChIP-seq/DNase-seq. Thus, the task in the challenge was designed by utilising chromatin feature annotations of Arabidopsis thaliana. This report focuses on the competition protocol, with our desired outcome being that experimental researchers will be able to hold competitions themselves in the future, from coordinating tasks to managing participant submissions, by following the procedures for competition implementation that we describe here.

OVERVIEW OF THE DDBJ CHALLENGE COMPETITION

Overview of the DDBJ Data Analysis Challenge competition The DDBJ Data Analysis Challenge (DDBJ Challenge) was a machine learning competition using the International Nucleotide Sequence Database, a large biomolecular database that exists as an international collaboration between DDBJ in Japan, EMBL/EBI in Europe and NCBI in the U.S.A. The competition was held during the 2016 Japanese school summer period, from July 6 to August 31, for the purpose of quantitatively evaluating the effects of crowdsourcing a data science project. We opened a website dedicated to the DDBJ Challenge (https://www.ddbj.nig.ac.jp/activities/ddbj-challenge-e.html) within the framework of the larger DDBJ website, and supported the competition with the high-performance computational resources of the NIG supercomputer.

In the competition, participants submitted prediction outputs using their machine learning models for a prediction task, automatically annotating DNA sequences into chromatin features. Two common types of machine learning competition tasks are prediction problems and insight problems (Kohavi et al., 2000). Solutions to prediction problems were previously known, which meant they could be utilised for model evaluation. However, solutions to insight problems were not known before the competition and thus the evaluation cost for these problems tended to be high. Therefore, we selected prediction problems for this competition.

A total of 38 participants took part in the ‘Predicting chromatin features from DNA sequences’ challenge task, with a cumulative total of 360 model submissions. Three top prize winners and a student prize winner were selected, with the student prize winner (top rank of all student participants) corresponding to fifth place overall. Out of the top five-ranking participants, only one (in fourth place) had an information science background. The other participants had bioinformat-
ics backgrounds, such as company workers and research scientists. In the following sections, we will present the detailed results from the DDBJ Challenge.

**Data analysis challenge task with biological features** The task involved in the challenge was to predict chromatin annotations from DNA fragment sequences of *A. thaliana* based on the protocol of the ChIP-Atlas database (Oki et al., 2018). The original ChIP-Atlas database contains peak-call annotations from the next-generation sequencing of five model organisms (human, mouse, fruit fly, nematode and budding yeast). The analytical conditions were ‘ChIP-seq’ and ‘DNase-Hypersensitivity’ from LIBRARY_STRATEGY and ‘illumina’ from PLATFORM, where both parameters were metadata attributes of the SRA database. As for human datasets, the SRA database included datasets from big analytical projects such as the ENCODE project (The ENCODE Project Consortium, 2012) and the Roadmap Epigenomics project (Bernstein et al., 2010).

We generated 163 new *A. thaliana* datasets for the competition task by analysing the SRA database under the ChIP-Atlas protocol. The versions of the analytical tools are summarised in Supplementary Table S1. From these, datasets with only a few peaks were discarded, and then eight datasets from two classes of Antigen and Cell Type were selected based on an informative number of annotated peaks (Table 1). Antigen classes can be specified by peak types of DNase-seq, ChIP-seq to characterise binding sites of transcription factors, ChIP-seq for identifying histone modification sites and others. The eight class combination labels were masked to prevent cheating by participants, and we documented these to prohibit any reannotation of the SRA *Arabidopsis* database.

The following training datasets for machine learning models were generated from the TAIR10 database of the *Arabidopsis* genome and annotations (https://abrc.osu.edu/).

- Input training data: 60,000 DNA fragment sequences
- Input test data: 10,000 DNA fragment sequences
- Output training data: 60,000 rows \( \times \) 8 conditions (True/False boolean labels)

A true prediction against test data of 10,000 rows with 8 conditions was requested for submission to the UniversityOfBigData website.

**Ethical approval and consent to participate** The DDBJ Challenge crowdsourcing research was approved by the Institutional Review Board (IRB) (Graber and Graber, 2013) of the National Institute of Genetics, Japan in March 2016. Informed consent with IRB approval was displayed to the crowd participants in the DDBJ Challenge and user agreement was obtained prior to users being registered on the UniversityOfBigData data science infrastructure of Kyoto University.

**Concealing life science domain-specific descriptions in the competition task** Predicted labels for training data comprise boolean code with one binary digit. True (1) or False (0) indicates that the DNA sequence contains or does not contain the characteristic chromatin region, respectively. This boolean description is popular in the domain of data science, while the description of DNA sequences was unfamiliar to the participants with no life science background. Thus, we converted nucleotide sequences with nucleotide letters requiring specific domain knowledge to zero-one-hot encoding vectors using binary digits to make them easier to handle for participants without domain-specific knowledge. Therefore, sequence fragments comprising 200 nucleotide letter codes were converted to 800 binary codes.

**Participant recruitment and submission management of prediction outputs** Participants were recruited through DDBJ newsletters and various email mailing lists from bioinformatics communities in

| SRA accession | LIBRARY_STRATEGY | Antigen class | Antigen | Cell type |
|---------------|------------------|---------------|---------|-----------|
| SRX9391997    | DNase-seq        | DNase-seq     | –       | flower    |
| SRX939193     | DNase-seq        | DNase-seq     | –       | seed      |
| SRX022326     | ChIP-Seq         | Histone       | H3K9ac  | rosette leaf |
| SRX346160     | ChIP-Seq         | Histone       | H3K4me3 | leaf      |
| SRX145429     | ChIP-Seq         | Histone       | H3K9me2 | leaf      |
| SRX156079     | ChIP-Seq         | RNA polymerase| RNA polymerase V | seedling |
| SRX128177     | ChIP-Seq         | TF            | PRR5    | whole plant |
| SRX159029     | ChIP-Seq         | TF            | MYC     | seedling  |

TF: Transcription factor.
Japan. The prediction results of chromatin feature annotations from the test DNA sequences used as input data were submitted using Kyoto University's educational data science platform, UniversityOfBigData. The competition dataset was distributed from two data sites, (1) the web server of the UniversityOfBigData, and (2) disk space on the NIG Supercomputer (Ogasawara et al., 2013). After submission, half of the predicted output data were utilised for computing tentative participant ranks based on the evaluation scores of the receiver operating curve–area under the curve (ROC–AUC). The remaining half of the dataset was retained to calculate the final scores (Supplementary Table S2).

Supporting GPU computational resources and tool installation requests To construct machine learning models, participants need to prepare their computational resources. The DDBJ Challenge committee supported 16 GPU nodes, part of the computing resources of the NIG supercomputer, where the special usage of GPU nodes expired during the final day of the competition. Moreover, user requests for open source software installation were accepted. Thus, data science tools such as BVLC Caffe (Jia et al., 2014) and PFN Chainer (Tokui et al., 2015) were installed on the NIG supercomputer. From Mathworks Japan, MATLAB licences were offered for the student competition during the DDBJ Challenge period.

DDBJ CHALLENGE AWARDS AND PARTICIPATORY FEATURES

Outline of the first-place model The first-place model in the DDBJ Challenge was constructed by Dr. Masahiro Mochizuki. He designed a compositional architecture model based on two classifiers of extremely randomized trees (ERT) (Geurts et al., 2006) and convolutional neural networks (CNNs) (LeCun et al., 1998). A stacking generalisation algorithm (Wolpert, 1992) was then applied for ensemble learning to integrate the two classifier models. He also utilised two external parameters, genomic coordination and gene annotation information, in addition to the default features of the challenge query sequences. The first ERT model was constructed as an $N \times M$ matrix, with $N$ as the index of input query sequences and $M$ as the chromosome number. The genomic coordinate position was computed by alignment analysis of query sequences against the TAIR10 Arabidopsis reference genome sequence. We will refer to this genomic coordinate-based ERT model as the genomic coordinates based model (GCBM). The second component of the model, the CNN, incorporated query DNA sequences and gene structural annotation information, where gene annotation data were downloaded as a general feature format (Eilbeck et al., 2005) file from the TAIR10 database. As shown in Fig. 1A, gene structural annotation was modelled by feature matrix representation with two parameters as follows:

$$(1 - r)^d$$ if the base is included in a gene of the strand,

Otherwise 0.

The variable $r$ is the attenuation rate and the variable $d$ is the distance of a genomic coordinate from the gene start coordinate. For instance, if the variable $r$ is zero, the feature is equal to 1, which indicates that the target coordinate position is included in an annotated gene coordinate. Likewise, when the variable $r$ is greater than zero, the gradient value from the gene start coordinate is computed. This CNN model based on gene annotation is called the gene annotated sequences based model (GASBM), indicating the effect of domain knowledge. Figure 1B depicts a schematic explanation of the CNN from feature matrix input to chromatin feature outputs. Figure 1C displays the benchmark results of the algorithms utilised in the first-place model. The performance of the individual GCBM and GASBM models was evaluated using their AUC evaluation scores. The query sequence-based model without gene annotation data was compared as a control.

Model performances and algorithms of the top three ranking models After the close of the DDBJ Challenge submission period, the top three overall winners and the student prize winner were asked to submit reports explaining the detailed structures of their proposed machine learning models. Table 2 summarises the performance and model algorithm designs based on these reports, including information on individual analytical tools. All of the top three winners used CNNs, and all deployed ensemble learning as a strategy to incorporate multiple algorithms. External parameters were incorporated by the first- and second-place winners. The first-place winner used two external parameters, genomic position and gene annotation with strand information, as stated above, while the second-place winner used only genomic position. The performance of the top model resulted in an AUC score of 0.95, where the performance of the first submitted model was 0.65. The value of the first submitted model was equivalent to the performance score of our tutorial with the linear discriminant analysis algorithm. Over the course of the competition, the overall performance of the submitted models improved by an AUC score of 0.30 from the first submitted model.

Incorporating domain knowledge into prediction models and the participants’ background knowledge of life science In general, domain knowledge is known to significantly raise performance scores in domain-specific machine learning tasks. For example, a study aimed at predicting the risk of breast cancer
Meeting report of a machine learning competition

found that the effect of domain knowledge of SNPs was a 6%–8% increase in accuracy (Bochare et al., 2014). Both the first- and second-place models incorporated external parameters as domain knowledge. From Table 2, the differences in the AUC scores between the models with domain knowledge (first and second place) and without domain knowledge (third place) were 0.092 and 0.045, respectively. The rough effect of domain knowledge, 5%–9%, is close to that found by Bochare et al. in the above-mentioned study. Furthermore, assessing the

Table 2. Model performances, designs and tools of the top prize winners

| Rank | Model performance (AUC) | Model design | Programming tool |
|------|-------------------------|--------------|------------------|
| 1st  | 0.946                   | - 2 classifiers (CNN, ERT) | Python 3.5       |
|      |                         | - Ensemble learning (Stacking) | Scikit-learn 0.17.1 |
|      |                         | - External data (Genomic position, Gene structural annotation) | Chainer 1.10.0 |
| 2nd  | 0.899                   | - 2 classifiers (CNN, Product of genomic distance decay parameter and nearest training data output) | Julia 0.4.6 |
|      |                         | - Ensemble learning (Averaged) | Python 2.7.10 |
|      |                         | - External data (Genomic position) | TensorFlow 0.8.0 |
| 3rd  | 0.854                   | - 7 classifiers (CNN, Logistic regression, Gradient boosting, Naive Bayes for multivariate Bernoulli models, Random forest, Extremely randomized trees, Extreme gradient boosting) | Python 2.7.11 |
|      |                         | - Ensemble learning (Stacking) | NumPy 1.10.4 |
|      |                         |                          | Scikit-learn 0.17 |
|      |                         |                          | Chainer 1.11.0 |
|      |                         |                          | XGBoost 0.4a30 |

Fig. 1. Overview of the first-place model in the DDBJ Data Analysis Challenge. (A) A schematic view of feature matrices with both strands representing a gene annotated sequence of the GASBM model. (B) A schematic view of the neural network structures of the GASBM model. (C) A benchmark result of the GASBM, GCBM and ensemble algorithms in the first-place model.
effect of domain knowledge using only the first-place model under the average of 8 conditions, we can confirm three differences in AUC scores as 0.034, −0.034 and 0.091 between the sequence-based model (Fig. 1C(a)) and the three respective models incorporating patterns, GASBM (Fig. 1C(c)), GCBM (Fig. 1C(d)) and the stacking model (Fig. 1C(e)), respectively. Regarding the participants’ background knowledge, the top three winners had background knowledge of life science, while the fourth-ranking participant had a background in information science. In the UniversityOfBigData platform, the DDBJ Challenge was the first life science domain task, which meant that the previous seven tasks, such as ‘Business Card OCR’, did not require life science knowledge. This domain knowledge effect might imply some barriers for participants from non-life science backgrounds when it comes to tackling life science machine learning competitions.

**Programming tools of top-ranking models** As shown in Table 2, all the winning models used the Python programming language, with its plentiful machine learning libraries, including scikit-learn (Pedregosa et al., 2011), and the deep learning libraries of Google's TensorFlow (Abadi et al., 2015) and PFN’s Chainer. The student prize winner also utilised Python 2.7 and the Lasagne package (Dieleman et al., 2015) as a wrapper library for the University of Montreal’s Theano (The Theano Development Team, 2016).

**Analysis of participatory metrics 1: Participation inequality** Participatory metrics analysis may be valuable for readers who want to plan machine learning competitions. We counted submission statistics using functions of the UniversityOfBigData platform. Figure 2A displays the pattern of submissions by participants, with the horizontal axis showing the days of the competition period and the vertical axis showing the intermediate prediction score of the competitions. In the field of crowdsourcing research, there is typically inequality in the number of submissions between the participants (Ortega et al., 2008), in that a small number of participants tend to account for a large number of submissions, as a fraction of the whole submission volume (Sauermann and Franzoni, 2015). Such participation inequality can be quantified by the Gini coefficient (Yang et al., 2016), which is often used in economics. The Gini coefficient, $G$, is defined as a ratio of the areas on the Lorenz curve diagram. If the area between the line of perfect equality and the Lorenz curve is denoted as $A$, and the area under the Lorenz curve is denoted as $B$, then the Gini coefficient can be calculated as $G = A/(A + B)$. The Gini coefficient is a global index, where 0 represents complete equality and 100 complete inequality. The Gini coefficient of submissions in our competition was 0.48, while those of other competition tasks in the UniversityOfBigData platform ranged from 0.43 to 0.54.

![Fig. 2](image-url) References for the participatory metrics analyses. (A) The pattern of submissions by participants, with the horizontal axis showing the days of the competition period and the vertical axis showing the intermediate prediction score of the competition. (B) Pearson’s correlation coefficient between the final performance scores (vertical axis) and the number of model submissions (horizontal axis) on the participants’ leaderboard activity.
Analysis of participatory metrics 2: Correlation between leaderboard activity and performance

Pearson’s correlation coefficient between the final performance scores and the number of submissions, as the participants’ leaderboard activity, was 0.35 ($P = 0.03$), which was a weak positive correlation, as shown in Fig. 2B. In crowdsourcing research, it has been reported that engagement score and submission number are not correlated based on annotation tasks (Good et al., 2015). As for crowdsourced machine learning modelling tasks, an online article discussing Kaggle competitions states that there is an association between making a greater total number of submissions and an increase in final ranking (https://rpubs.com/pedmiston/kaggle). However, another study (Küffner et al., 2015) reported unclear results, with both positive and negative correlations between the number of submissions and final performance score depending on the datasets used. Concerning the correlation analysis of performance score with the number of submissions, it may be necessary to collect data on further tasks.

CONCLUDING REMARKS

We described the DDBJ Data Analysis Challenge, a machine learning competition in the life science domain held during the 2016 summer period. This report will provide reference knowledge for ensuing challengers and task planners, and encourage future machine learning competitions in life science domains.

We thank all of the participants in the competition: orion, Ryota, tsukasa, emn, extraterrestrial Fuu species, ñzw, mkoido, ksh, AoYu@Tohoku, hiro, emihat, MorikawaH, hmt-yamamoto, tonets, suzudora, take2, bicycle1885, morizo, forester, doiyasan, yudai, tag, nwatarai, soki, tsunechan, Ken, A.K, singular0316, IK, yk_tani, yota0000. We are also grateful to Ayako Oka, Yasuhiro Tanizawa, Takako Mochizuki, Fumi Hayashi, Naoko Sakamoto and Tarzo Ohta for their support in preparing the datasets for the task, and to Fumitaka Otobe, Takuya Ohtani, Hijari Amano, Takafumi Ohbiraki, Yuji Ashizawa, Tomohiko Yasuda, Naofumi Ishikawa, Tomohiro Hirai, Tomoka Watanabe, Chiharu Kawagoe, Emi Yokoyama, Kimiko Suzuki and Junko Kohira for their computational infrastructure and management support. Data analysis was partially performed using the Research Organization of Information and Standards (ROIS) NIG Supercomputer System. This research was partially supported by management expenses grants from the DNA Data Bank of Japan, the ROIS, and JST CREST Grant Number JPMJCR1501, Japan.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2015) TensorFlow: Large-scale machine learning on heterogeneous distributed systems. Software available from tensorflow.org. arXiv:1603.04467.
Baba, Y., Takase, T., Atarashi, K., Oyama, S., and Kashima, H. (2018) Data analysis competition platform for educational purposes: lessons learned and future challenges. Proceedings of the 8th Symposium on Educational Advances in Artificial Intelligence 2018 (EAAI-18), 7887–7892.
Bender, E. (2016) Challenges: crowdsourced solutions. Nature 533, S62–S64.
Bernstein, B. E., Stamatoyannopoulos, J. A., Costello, J. F., Ren, B., Milosavljevic, A., Meissner, A., Kellis, M., Marra, M. A., Beaudet, A. L., Ecker, J. R., et al. (2010) The NIH roadmap epigenomics mapping consortium. Nat. Biotechnol. 28, 1045–1048.
Bochare, A., Gangopadhyay, A., Yesha, Y., Joshi, A., Yesha, Y., Brady, M., Grasso, M. A., and Rishe, N. (2014) Integrating domain knowledge in supervised machine learning to assess the risk of breast cancer. Int. J. Med. Eng. Inform. 6, 87–99.
Brinkmann, B. H., Wagenaar, J., Abbot, D., Adkins, P., Bossard, S. C., Chen, M., Tieng, Q. M., He, J., Muñoz-Almaraz, F. J., Botella-Rocamora, P., et al. (2016) Crowdsourcing reproducible seizure forecasting in human and canine epilepsy. Brain 139, 1713–1722.
Cochrane, G., Karchs-Mizrachi, I., Takagi, T., and International Nucleotide Sequence Database Collaboration (2016) The international nucleotide sequence database collaboration. Nucleic Acids Res. 44, D48–D50.
Dieleman, S., Schlüter, J., Raffel, C., Olson, E., Sønderby, S. K., Nouri, D., Maturana, D., Thoma, M., Battenberg, E., Kelly, J., et al. (2015) Lasagne: First release. doi:10.5281/zenodo.27678.
Eilbeck, K., Lewis, S. E., Mungall, C. J., Yandell, M., Stein, L., Durbin, R., and Ashburner, M. (2005) The Sequence Ontology: a tool for the unification of genome annotations. Genome Biol. 6, R44.
Friedberg, I., Wass, M. N., Mooney, S. D., and Radivojac, P. (2015) Ten simple rules for a community computational challenge. PLoS Comput. Biol. 11, e1004150.
Geurts, P., Ernst, D., and Wehenkel, L. (2006) Extremely random trees. Mach. Learn. 63, 3–42.
Good, B. M., Nanis, M., Wu, C., and Su, A. I. (2015) Microtask crowdsourcing for disease mention annotation in PubMed abstracts. Pac. Symp. Biocomput. 282–293.
Graber, M. A., and Graber, A. (2013) Internet-based crowdsourcing and research ethics: the case for IRB review. J. Med. Ethics 39, 115–118.
Hirschman, L., Yeh, A., Blaschke, C., and Valencia, A. (2005) Overview of BioCreAtIvE: critical assessment of information extraction for biology. BMC Bioinformatics 6, S1.
Janin, J. (2005) Assessing predictions of protein-protein interaction: the CAPRI experiment. Protein Sci. 14, 278–283.
Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014) Caffe: Convolutional architecture for fast feature embedding. arXiv:1408.5903.
Kodama, Y., Mashima, J., Kosuge, T., Kaminuma, E., Ogasawara, O., Okubo, K., Nakamura, Y., and Takagi, T. (2018) DNA Data Bank of Japan: 30th anniversary. Nucleic Acids Res. 46, D30–D35.
Kodama, Y., Shumway, M., and Leinonen, R. (2012) The sequence read archive: explosive growth of sequencing data. Nucleic Acids Res. 40, D54–D56.
Kohavi, R., Brodley, C. E., Frasca, B., Mason, L., and Zheng, Z. (2000) Kdd-cup 2000 organizers’ report: peeling the onion. ACM SIGKDD Explor. Newsl. 2, 86–92.
Küffner, R., Zach, N., Norel, R., Hawe, J., Schoenfeld, D., Wang, L., Li, G., Fang, L., Mackey, L., Hardiman, O., et al. (2015)
Crowdsourced analysis of clinical trial data to predict amyotrophic lateral sclerosis progression. Nat. Biotechnol. 33, 51–57.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998) Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324.

Meyer, P., Alexopoulos, L. G., Bonk, T., Califano, A., Cho, C. R., de la Fuente, A., de Graaf, D., Hartemink, A. J., Hoeng, J., Ivanov, N. V., et al. (2011) Verification of systems biology research in the age of collaborative competition. Nat. Biotechnol. 29, 811–815.

Moult, J. (2015) A decade of CASP: progress, bottlenecks and prognosis in protein structure prediction. Curr. Opin. Struct. Biol. 15, 285–289.

Ogasawara, O., Mashima, J., Kodama, Y., Kaminuma, E., Nakamura, Y., Okubo, K., and Takagi, T. (2013) DDBJ new system and service refactoring. Nucleic Acids Res. 41, D25–D29.

Oki, S., Ohta, T., Shioi, G., Hatanaka, H., Ogasawara, O., Okuda, Y., Kawaji, H., Nakaki, R., Sese, J., and Meno, C. (2018) ChIP-Atlas: a data-mining suite powered by full integration of public ChIP-seq data. EMBO Rep. 19, e46255.

Ortega, F., Gonzalez-Barahona, J. M., and Robles, G. (2008) On the inequality of contributions to Wikipedia. Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008), 304.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011) Scikit-learn: machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.

Prill, R. J., Saez-Rodriguez, J., Alexopoulos, L. G., Sorger, P. K., and Stolovitzky, G. (2011) Crowdsourcing network inference: the DREAM predictive signaling network challenge. Sci. Signal. 4, mr7.

Saez-Rodriguez, J., Costello, J. C., Friend, S. H., Kellen, M. R., Mangravite, L., Meyer, P., Norman, T., and Stolovitzky, G. (2016) Crowdsourcing biomedical research: leveraging communities as innovation engines. Nat. Rev. Genet. 17, 470–486

Sauer, T., and Franzoni, C. (2015) Crowd science user contribution patterns and their implications. Proc. Natl. Acad. Sci. USA 112, 679–684.

The ENCODE Project Consortium (2012) An integrated encyclopedia of DNA elements in the human genome. Nature 489, 57–74.

The Theano Development Team (2016) Theano: A Python framework for fast computation of mathematical expressions. arXiv:1605.02688.

Tokui, S., Oono, K., Hido, S., and Clayton, J. (2015) Chainer: a next-generation open source framework for deep learning. Proceedings of Workshop on Machine Learning Systems in the 29th Annual Conference on Neural Information Processing Systems (NIPS) 5, 1–6.

van Nimwegen, K. J., van Soest, R. A., Veltman, J. A., Nelen M. R., van der Wilt, G. J., Vissers L. E. L. M., and Grutters, J. P. C. (2016) Is the $1000 genome as near as we think? A cost analysis of next-generation sequencing. Clin. Chem. 62, 1458–1464.

Wazny, K. (2017) “Crowdsourcing” ten years in: a review. J. Glob. Health 7, 020602.

Wolpert, D. H. (1992) Stacked generalization. Neural Netw. 5, 241–259.

Yang, A., Fan, H., Jing, N., Sun, Y., and Zipf, A. (2016) Temporal analysis on contribution inequality in openstreetmap: a comparative study for four countries. Int. J. Geo-inf. 5, 5.
Supplementary Information

Table of contents for supplementary information

**Supplementary Note S1** – Summary of the first-place model from the DDBJ Data Analysis Challenge

**Supplementary Note S2** – Summary of the second-place model from the DDBJ Data Analysis Challenge

**Supplementary Note S3** – Summary of the third-place model from the DDBJ Data Analysis Challenge

**Supplementary Table S1** – List of tool versions based on the same analytical protocol of ChIP-Atlas database.

**Supplementary Table S2** – Summarising the performances and activity of participants on the leaderboard of the UniversityOfBigData system.

**Supplementary Table S3** – Hyperparameters of the neural network used in GASBM

**Supplementary Table S4** – Hyperparameters of a combiner of stacking
Note S1. Summary of the first-place model from the DDBJ Data Analysis Challenge

A prediction model with genomic coordinates used as features was built, called the genomic coordinates based model (GCBM). As genomic coordinates of each sequence in datasets had not been provided by the organisers, the sequences were mapped to the *Arabidopsis thaliana* genome (TAIR10). Feature matrices for training and test datasets were generated from this mapping result. In particular, an $n \times m$ matrix was initially set to zeros, where $n$ is the number of sequences and $m$ is the number of chromosomes. Then, the position of the 5' end base of each sequence was set to the corresponding element. An extremely randomized tree (ERT) classifier was trained on the feature matrix for the training dataset to predict whether each of eight chromatin features was contained in each sequence in the test dataset. Hyperparameters of the ERT classifier used in GCBM were estimated as $\text{bootstrap}=\text{TRUE}$, $n_{\text{estimators}}=3000$, and $\text{max\_features}=1$ using the constructor of ExtraTreesClassifier class of scikit-learn version 0.17.1. Moreover, another model, namely the gene annotated sequences based model (GASBM), was added. This model is basically a convolutional neural network (CNN) like DeepBind (Alipanahi et al., 2015), partly modified to adjust to the problem of this competition and improve prediction performance. For instance, a sigmoid function instead of a softmax function was adopted in the output layer, and two types of kernel sizes were used together in the convolutional layer of a CNN model for sentiment analysis (Johnson and Zhang, 2015). Batch normalization (Ioffe and Szegedy, 2015) was applied to the convolutional layer to accelerate training. Adam (Kingma and Ba, 2015) was used as an optimiser with default parameters. Figure 1B and Supplementary Table S3 show a schematic of the network and the hyperparameters, respectively. In addition, while DeepBind inputs a $k \times 4$ matrix for a $k$ base pairs (bp) sequence, where the four columns of the matrix correspond to each of the four bases, in the first-place model, a matrix for input has two additional columns representing gene annotation information of both strands. In particular, a value in the columns is determined by two parameters of the decay rate $r$ and the distance from the first base of the gene $d$, as described in the section of ‘Outline of the first-place model’. Therefore, the columns indicate whether the base is contained in a certain gene. When $r$ is greater than 0, the columns form gradients from the start to the end of the gene. These gradients inform not only whether the base is inside a certain gene but also the relative distance of each base in the gene from the starting point of the gene. An example of the input matrix is shown in Fig. 1A. The positional information of genes was based on GFF files downloaded from the website of The *Arabidopsis* Information Resource. Both original sequences and reverse complementary sequences were used for training. Predictions were performed on both of the original and reverse complementary sequences. The average between them was adopted as the final prediction.

Finally, the two prediction models, GCBM (decay rate = 0.001) and GASBM, were combined by means of stacking in order to improve the accuracy of the final prediction. Ten-fold cross validation
of the two models was performed to derive the meta-features for training. A multi-layer perceptron with ReLU activation functions was used as a combiner of stacking. The hyperparameters of the network are shown in Supplementary Table S4. To benchmark the performances of the individual and stacked models, the training dataset provided by the organiser was randomly divided into two groups: a training dataset (54,000 sequences) and a test dataset (6,000 sequences). Each of the models was trained on the former and tested on the latter. In addition to GASBM and GCBM, a sequence-based model without gene annotation was tested. This model is the same as GASBM except that it inputs four column matrices like DeepBind. The result of the benchmarking is shown in Fig. 1C. The best model among the sequence-based models was the GASBM with a decay rate = 0.001. Furthermore, a stacked model between GCBM and GASBM outperformed both of the individual models on average, and even on each of the chromatin features. Finally, the stacked model was newly trained on the whole of the training dataset (60,000 sequences) once again to predict chromatin features on the real test dataset (10,000 sequences) and submit the result of the prediction. As a result, an ROC–AUC score of 0.94564 was achieved as the final score of the competition.

This method has the following notable points: construction of a model based on the genomic coordinates of each DNA fragment, and the introduction of gene annotation information to a conventional sequence-based CNN model. The former is founded on the simple idea that adjacent regions on a chromosome will present similar chromatin features. Although the GCBM model was inferior to the other models on average as shown in Fig. 1C(d), it was superior to them on chromatin features 5 and 7. This suggests that a part of chromatin features depends on the environment surrounding a region of genomic DNA rather than the sequence of the region. With regard to the latter point, it was found that gene annotation is informative for predicting chromatin features from a sequence. Moreover, introducing gradients to features generated by gene annotation further improved the performance of the model. This result is consistent with the idea that the gradient enabled a classifier to know the distance of each base in some gene from the start point of the gene as described above. However, the concrete mechanism of contribution by gene annotation requires further investigation. At the same time, there is probably still room for optimisation on the utilisation of gene annotation, e.g., whether a gradient should be oriented upstream, downstream as in this case, or both, and whether or not the exponential decay used is the optimal curve. As mentioned above, the areas of specialty of each model were different from each other. Thus, the models effectively complement their respective disadvantages. Indeed, as a result of stacking between the two models, the prediction performance on all eight chromatin features was considerably enhanced.

References
Alipanahi, B., Delong, A., Weirauch, M. T., and Frey, B. J. (2015) Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. Nat. Biotechnol. 33, 831–838.
Ioffe, S., and Szegedy, C. (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. Proceedings of the 32nd International Conference on Machine Learning (PMLR) 37, 448–456.

Johnson, R., and Zhang, T. (2015) Effective use of word order for text categorization with convolutional neural networks. arXiv:1412.1058.

Kingma, D. P., and Ba, J. (2015) Adam: A method for stochastic optimization. Proceedings of the 3rd International Conference for Learning Representations (ICLR), 1–15.
Note S2. Summary of the second-place model from the DDBJ Data Analysis Challenge

First prediction model
The sequences in the training and test data were converted into FASTA format using a custom Julia script. First, the genomic location of each sequence was searched in the training and test data to investigate the spatial distribution of the sequences. The sequences were mapped to the *A. thaliana* genome (TAIR10) using Bowtie2 (Langmead and Salzberg, 2012) (version 2.2.8) with the parameters ‘--mp 1000,1000 --rdg 1000,1000 --rfg 1000,1000 -f -U’. The resulting BAM files were converted to BED files using the ‘bamtobed’ command of BEDTools (Quinlan and Hall, 2010) (version 2.17.0). Then, for each test sequence, the distance between a test sequence and its closest training sequence was calculated using the ‘closest’ command of BEDTools with the options ‘-t first -d’. The second-place winner refers the distances to ‘closest distance’. If a test sequence was not aligned to the genome or did not have any closest training sequence on the same chromosome, the closest distance was set to the missing value. It was found that the closest distance showed peaks at every 200 bp (1, 201, 401, 601,…), and 83.2% of the test sequences had a closest distance of no more than 401 bp. Based on these observations, to predict target variables of each test sequence, only target variables of its closest training sequence were considered. A weight $w$ was set as decaying according to the closest distance. When the closest distance was $d$ bp, $w$ was calculated as follows: $w = \exp(-d/d0)$. Then $d0$ was set to 1000 bp in the following analysis. Note that if a test sequence was not aligned to the genome or did not have any closest training sequence, the weight was set to zero. For each test sequence and each target variable, the prediction was the product of $w$ and the target variable value (0 or 1) of the closest training data.

Second prediction model
The original combinatorial target variables were regarded as eight independent target variables, and the following processes were implemented for each target variable. Because there were more positive than negative data, the negative data were reduced randomly so that the ratio of positive data to negative data was approximately 1.0. Each 200-nucleotide sequence was converted into $k$-mer count vectors ($k = 2, 3, 4$ and 5). For each $k$, a following predictive model was developed independently. A deep learning algorithm was used to predict two classes (0 or 1). The ‘skflow’ library was used for implementation. The hyperparameters were set for $k = 2, 3, 4$ and 5 (\textit{hidden_units} = 10, \textit{batch_size} = 128, \textit{steps} = 2000, and \textit{learning_rate} = 0.05). A predictive model was also developed with different hyperparameters for $k = 5$ (\textit{hidden_units} = 50, \textit{batch_size} = 128, \textit{steps} = 5000, \textit{learning_rate} = 0.05).
Integration of the two prediction models

The first and second prediction models were designed to predict positive values with high sensitivity and specificity, respectively. To avoid overfitting, the learning process was discontinued at an early stage of parameter optimisation. The two classes for test data were predicted with the above five learned models, separately. The final prediction was created by averaging the five independent predictions. Virtual test data were also constructed from the training data and the performance of the above method was evaluated. The predictions were averaged from the first and second methods. The AUC value of the integrative method was 0.89859, which is higher than that of each method independently.

Epigenetic modifications generally span regions of several hundred to thousands of base pairs, and show gradual spatial transitions along the genome. Such characteristics have prompted researchers to use probabilistic models for sequential data (e.g., the hidden Markov model (Ernst and Kellis, 2012) and the dynamic Bayesian network (Hoffman et al., 2012)) for learning ‘chromatin states’ from model epigenetic modifications. Thus, it is natural to make use of spatial information to predict epigenetic modifications of a given sequence extracted from a chromosome. Indeed, after incorporating predictions from spatial information, the AUC showed improvement. This indicates that spatial information as well as sequence features contain complementary information for predicting epigenetic modifications.

References

Ernst, J., and Kellis, M. (2012) ChromHMM: automating chromatin-state discovery and characterization. Nat. Methods 9, 215–216.
Hoffman, M. M., Buske, O. J., Wang, J., Weng, Z., Birmes, J. A., and Noble, W. S. (2012) Unsupervised pattern discovery in human chromatin structure through genomic segmentation. Nat. Methods 9, 473–476.
Langmead, B., and Salzberg, S. L. (2012) Fast gapped-read alignment with Bowtie 2. Nat. Methods 9, 357–359.
Quinlan, A. R., and Hall, I. M. (2010) BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics 26, 841–842.
Note S3.  Summary of the third-place model from the DDBJ Data Analysis Challenge

Normal k-mer frequency, complementary shrunk k-mer frequency, and leaped k-mer frequency were adopted. There were a few varieties of frequency aggregation by frame window size. A single full frame of 200 bases is basic, and prior 100 bases, posterior 100 bases, and overwrapping mid 100 bases are also supplementarily adopted in some branches of training. The code ‘N’ is simply dropped out of counts of k-mer for clarity. Besides k-mer frequency, two other features were incorporated into the third-place model. One is sequence entropy, based on the frequency of each base code.

The basic pattern of the model training is fundamentally by basic stacking of k-folds. The list of classifiers for stacking is as follows, named conventionally by their Python function names in the module:

1)  BernoulliNB
2)  LogisticRegression
3)  RandomForestClassifier
4)  GradientBoostingClassifier
5)  ExtraTreeClassifier
6)  XGBClassifier
7)  Chainer

There may be some duplicate functions, but this may be tolerated because the stacking method, as an every-in and best-out algorithm, fixes the duplication. Regarding Chainer, with its specific access interface, a Python ‘scikit-learn’ classifier was implemented as a class-compatible wrapper to work under the stacking framework with ‘fit’ and ‘predict_proba’ functions.

Each prediction for each part in the stacking was mostly done by Python’s default Logistic regression function, except for one case. The exception was for the final blend function, which was performed by elastic net. This elastic net function is not for regression but for classification/discrimination, and was implemented to maximise log-likelihood with $L1$ and $L2$ regularisation constraints. An elastic net function was originally implemented, and it is written in Java language using the Newton-Raphson method for the calculation of coefficients. For this competition, a Python wrapper interface was implemented like a ‘scikit-learn’ classifier class to incorporate multiple functions as the total system. The $L1:L2$ ratio is an important factor for the total system performance, especially to avoid overfitting, and an empirical trial suggested that $L1:L2 = 10:1$ was optimum and not too sparse. The first trial to estimate the system performance was as follows: using a 2/3/4/5-mer frequency mix (16+64+256+1024 predictors), a total of 60,000 retrospective supervised samples had been trained to predict 10,000 prospective samples, simply by default logistic regression, yielding an AUC value of 0.80508. Upon this basic linear model, the final performance was expected to be improved by these strategies:
1) By application of the non-linear model and deep learning system Chainer.
2) By application of weak learner aggregation as boosting.
3) By application of XGBoost with the best reputation.
4) By application of stacking upon these classifiers.
5) By application of extra predictors as well as complement shrinkage.
6) By application of elastic net to the final blending of stacking.

Deep learning, XGBoost and stacking, in this order, were likely the major contributors to the improvement of AUC. Ultimately, the result reached the scores of 0.85562 (intermediate) and 0.85428 (final). Some of the stacking discriminators and their parameters were adopted ad hoc in the analysis. The key point with major importance is to avoid overfitting throughout the prediction system. In that sense, multiple applications of simple stacking may result in the information link between predictors and supervising signals. The computing environment was Xeon W5590@3.33GHz ubuntu 14.04 server, java1.7.0_51.

**Detailed workflow**

notations for predictors in each flow below:

- \( o5 = 5\text{-mer complementary shrunk frequency.} \)
- \( l234 = 2\text{-mer frequency with a leap of 2/3/4 bases among them.} \)
- \( r123 = \text{maximum span length of 1/2/3-mer repetition.} \)
- \( m6 = 6\text{-mer normal (forward only) strand frequency.} \)
- \( e3 = \text{entropy by 3 frames (0, 100), (50, 150), (100, 200).} \)
- \( m4f3 = 4\text{-mer frequency by 3 frames (0, 100), (50, 150), (100, 200).} \)
- \( m5f2 = 5\text{-mer frequency by 2 frames (0, 100), (100, 200).} \)

analysis flows:

- \( A: o5+l234+r123 \) (937 predictors); stacked (twice).
- \( B: \text{result of } A+m6 \) (4224 predictors); stacked (once).
- \( C: m4f3 \) (408 predictors); stacked (once).
- \( D: \text{result of } A, B, C \) (384 predictors); stacked (once).
- \( E: \text{result of } D+m5f2 \) (1152 predictors); stacked (once).
- \( F: \text{result of } A, B, C, D, E \) (640 predictors); final blend by elastic net.

parts of stacking classifiers:

1) BernoulliNB()
2) LogisticRegression(penalty='l2')
3) RandomForestClassifier(n_estimators=100, criterion='gini')
4) RandomForestClassifier\((n\_estimators=100, criterion='entropy')\)
5) GradientBoostingClassifier\((n\_estimators=25, random\_state=1)\)
6) GradientBoostingClassifier\((n\_estimators=30, random\_state=2)\)
7) ExtraTreesClassifier\((n\_estimators=100, criterion='gini')\)
8) XGBClassifier\((n\_estimators=100, max\_depth=3)\)
9) XGBClassifier\((n\_estimators=100, max\_depth=4)\)
10) XGBClassifier\((n\_estimators=300, max\_depth=3)\)
11) XGBClassifier\((n\_estimators=600, max\_depth=3)\)
12) XGBClassifier\((n\_estimators=1200, max\_depth=3)\)
13) ChainerClassifier\((n\_units=1000, hidden\_layers=1)\) (except \(A\))
14) ChainerClassifier\((n\_units=1000, hidden\_layers=2)\)
15) ChainerClassifier\((n\_units=1000, hidden\_layers=3)\) (except \(A\))

It also has to be noted that all stackings are by 3-fold using the simple Python KFold function. Rough confirmation was made that the distributions among the case and control of each of the eight signals were not too biased among the folded groups. Training of Chainer was conducted in permutated chunk sets of size 500 repetitively, with each forward process followed by back-propagation. The activation function for the rectifier in each node was ReLU. The numbers of nodes were all identical (1000) for the multiple layered models. These uniform simplifications empirically came from small sub-tests.
Supplementary Table S1.  List of tool versions based on the same analytical protocol of ChIP-Atlas database

|   | Tool name | Tool version | Tool URL                                |
|---|-----------|--------------|-----------------------------------------|
| 1 | Bowtie 2  | 2.2.8        | http://bowtie-bio.sourceforge.net/bowtie2/index.shtml |
| 2 | SAMtools  | 0.1.19       | http://samtools.sourceforge.net/       |
| 3 | MACS2     | 2.1.1.20160309 | https://github.com/taoliu/MACS/         |
Supplementary Table S2. Summarising the performances and activity of participants on the leaderboard of the UniversityOfBigData system

| Rank | Intermediate score | Final score | #Submissions |
|------|--------------------|-------------|--------------|
| 1    | 0.94403            | 0.94564     | 23           |
| 2    | 0.89694            | 0.89859     | 7            |
| 3    | 0.8562             | 0.85428     | 34           |
| 4    | 0.8514             | 0.84791     | 19           |
| 5    | 0.84548            | 0.84318     | 38           |
| 6    | 0.83687            | 0.83792     | 7            |
| 7    | 0.83375            | 0.83464     | 4            |
| 8    | 0.83578            | 0.83318     | 3            |
| 9    | 0.8282             | 0.82746     | 10           |
| 10   | 0.81045            | 0.81279     | 12           |
| 11   | 0.81506            | 0.81185     | 17           |
| 12   | 0.81087            | 0.8089      | 19           |
| 13   | 0.81436            | 0.80845     | 13           |
| 14   | 0.81027            | 0.80602     | 20           |
| 15   | 0.80469            | 0.80544     | 6            |
| 16   | 0.79833            | 0.79993     | 15           |
| 17   | 0.78738            | 0.79558     | 3            |
| 18   | 0.79107            | 0.78787     | 2            |
| 19   | 0.79088            | 0.78773     | 6            |
| 20   | 0.78476            | 0.78039     | 3            |
| 21   | 0.78045            | 0.77276     | 4            |
| 22   | 0.76901            | 0.76511     | 3            |
| 23   | 0.75673            | 0.76022     | 8            |
| 24   | 0.75549            | 0.75833     | 5            |
| 25   | 0.73422            | 0.73278     | 9            |
| 26   | 0.71853            | 0.71079     | 2            |
| 27   | 0.66085            | 0.66467     | 3            |
| 28   | 0.65021            | 0.65225     | 3            |
| 29   | 0.64769            | 0.64982     | 9            |
| 30   | 0.63322            | 0.6347      | 1            |
| 31   | 0.63322            | 0.6347      | 1            |
| 32   | 0.63322            | 0.6347      | 1            |
| 33   | 0.60364            | 0.60891     | 21           |
| 34   | 0.58843            | 0.58998     | 10           |
| 35   | 0.53703            | 0.52929     | 2            |
| 36   | 0.5035             | 0.50227     | 1            |
| 37   | 0.5                 | 0.5         | 10           |
| 38   | 0.16516            | 0.16702     | 6            |
### Supplementary Table S3. Hyperparameters of the neural network used in GASBM

**(A) Hyperparameters on the network structure**

| Layer                     | Hyperparameter      | Setting |
|---------------------------|---------------------|---------|
| Convolution layer 1       | Filter size         | 15      |
|                           | Number of filters   | 300     |
|                           | Stride              | 1       |
|                           | Ratio of dropout    | 0.6     |
| Convolution layer 2       | Filter size         | 6       |
|                           | Number of filters   | 300     |
|                           | Stride              | 1       |
|                           | Ratio of dropout    | 0.6     |
| Average Pooling layer     | Window size         | full length |
| Layers of MLP             | Number of units at 1st hidden layer | 1000 |
|                           | Number of units at 2nd hidden layer | 100 |
|                           | Ratio of dropout for each hidden layer | 0.5 |

**(B) Hyperparameters on others**

| Hyperparameter             | Setting |
|----------------------------|---------|
| Number of epochs           | 40      |
| Batch size                 | 50      |
Supplementary Table S4. Hyperparameters of a combiner of stacking

(A) Hyperparameters on the network structure

| Hyperparameter                      | Setting |
|-------------------------------------|---------|
| Number of units at 1st hidden layer | 1000    |
| Number of units at 2nd hidden layer | 1000    |
| Ratio of dropout for each layer     | 0.5     |

(B) Hyperparameters on others

| Hyperparameter | Setting |
|----------------|---------|
| Number of epochs | 40      |
| Batch size      | 100     |