Implementasi Prediksi Siswa *Dropout* pada MOOC Menggunakan Metode *Stacking Super Learner* dalam Lingkungan Komputasi Berkinerja Tinggi

Mery Yulinda Rahmi, Arif Djunaidy, dan Izzat Aulia Akbar
Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember (ITS)
e-mail: adjunaidy@its.ac.id

***Abstrak*—Permasalahan utama di berbagai platform MOOC (*massive open online course*), yaitu tingginya tingkat *dropout* yang bahkan dapat mencapai 91%–93%. Hal ini tentu berdampak terhadap profitabilitas bisnis MOOC. Oleh sebab itu, diperlukan model prediksi siswa *dropout* pada MOOC untuk memungkinkan adanya intervensi pencegahan *dropout*. Namun, besarnya ukuran data siswa MOOC membuat proses pemodelan tersebut memerlukan komputasi yang tinggi. Dengan melihat permasalahan tersebut, maka penelitian ini membangun model prediksi menggunakan metode *stacking* yang mutakhir, yakni *Super Learner*, dan dikomputasikan secara paralel menggunakan GPU atau CPU dalam lingkungan komputasi berkinerja tinggi. Pembelajar dasar yang menyusun model *Super Learner* meliputi *Logistic Regression*, KNN, SVM, *Naïve Bayes*, *Random Forest*, dan XGBoost, sedangkan *meta-learner* yang dieksperimenkan adalah NNloglik (*non-negative binomial likelihood maximization*) dan AUC-maxim (AUC *maximization*). Hasil eksperimen menunjukkan bahwa *Super Learner* dengan *meta-learner* AUC-maxim maupun NNloglik berhasil mengungguli kinerja model pembelajar dasar dan model yang menggunakan metode *stacking* lainnya, yaitu *Stacked Generalization*. Kedua model tersebut mencapai skor F1 secara berurutan sebesar 0,90139 dan 0,90126. Di samping itu, ditemukan bahwa paralelisasi GPU pada percobaan ini menghasilkan *speedup* komputasi hingga 2,4–23,3 kali lebih unggul daripada paralelisasi pada CPU.**

***Kata Kunci*—Komputasi Berkinerja Tinggi, MOOC *Dropout*, *Stacking*, *Super Learner*.**

I. PENDAHULUAN

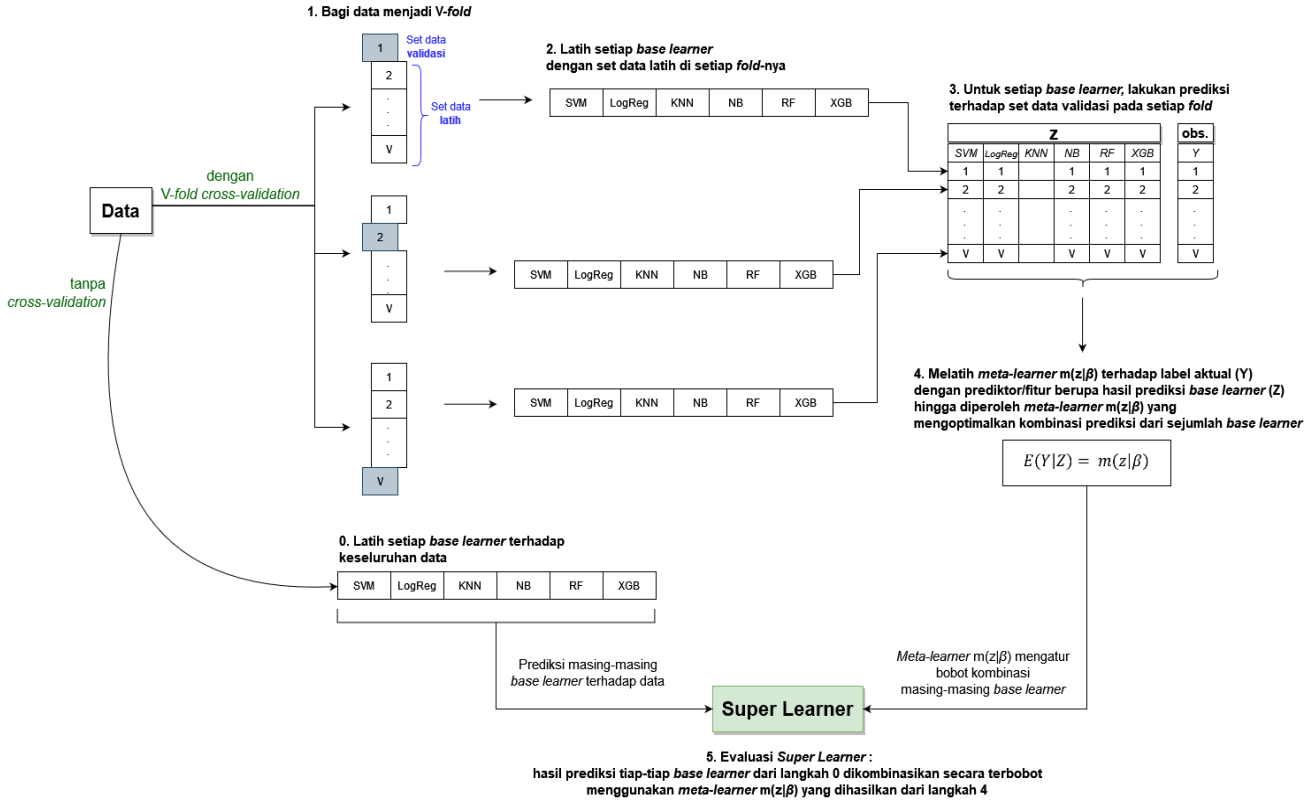
MOOC atau *massive open online course* menjadi salah satu bentuk edukasi daring yang kini semakin marak seiring dengan pesatnya perkembangan teknologi informasi [1]. Kehadiran MOOC menawarkan kegiatan pembelajaran yang fleksibel, baik dari segi waktu dan tempat, serta bersifat terbuka bagi siapa pun [2]. Akan tetapi, terlepas dari banyaknya siswa yang mendaftar kursus daring (*online course*), nyatanya sekitar 91%–93% siswa pada berbagai platform MOOC terbukti *dropout* atau tidak menyelesaikan pembelajaran kursusnya [3]. Hal ini tentunya memberi dampak terhadap profitabilitas bisnis MOOC sebab pendapatan utama model bisnis MOOC umumnya berasal dari biaya sertifikasi setelah siswa menyelesaikan kursusnya [4]. Akibatnya, tingginya tingkat *dropout* ini membuat bisnis MOOC berpotensi merugi. Salah satu upaya untuk mengurangi tingkat *dropout* pada MOOC adalah dengan pemodelan prediktif (*predictive modelling*) yang didasarkan pada data aktivitas siswa MOOC [5]. Namun, muncul tantangan lain, yaitu pemrosesan data MOOC juga bisa memakan waktu yang lama, mengingat besarnya volume data tersebut.

Pada penelitian sebelumnya, Xing *et al.* (2016) dan Youssef *et al*. (2019) telah mencoba membangun model prediksi siswa *dropout* pada MOOC menggunakan salah satu algoritme dari metode ansambel *stacking*, yaitu *Stacked Generalization* [6−7]. Penelitian Xing *et al*. (2016) mengimplementasikan *Stacked Generalization* menggunakan algoritme *Bayesian Network* dan *Decision Tree* (DT) sebagai pembelajar dasar (*base learner*) [6]. Sementara itu, Youssef *et al.* (2019) menggunakan pembelajar dasar yang lebih beragam, yaitu *Logistic Regression* (LogReg), DT, *Support Vector Machine* (SVM), *K-Nearest Neighbor* (KNN), *Naïve Bayes* (NB), dan *Random Forest* (RF) dan menggunakan *metalearner* berupa LogReg [7]. Kedua penelitian tersebut berhasil menunjukkan bahwa model yang menggunakan *Stacked Generalization* mempunyai keakuratan prediksi yang lebih baik dibandingkan hanya menggunakan masing-masing pembelajar dasarnya secara individu.

Akan tetapi, terdapat algoritme *stacking* yang lebih mutakhir daripada *Stacked Generalization*, yaitu *Super Learner* yang mengombinasikan pembelajar dasar dengan dibobotkan secara optimal terhadap fungsi kerugian (*loss function*) atau fungsi obyektif tertentu. Dalam beberapa penelitian pada domain lain, telah ditunjukkan bahwa *Super Learner* menghasilkan prediksi dengan keakuratan yang lebih baik daripada model *Stacked Generalization* [8–9]. Oleh sebab itu, penelitian ini membangun model prediksi *dropout* menggunakan algoritme *Super Learner* yang terdiri atas pembelajar dasar berupa LogReg, KNN, SVM, NB, RF, dan *Extreme Gradient Boosting* (XGBoost). Sementara itu, eksperimen terhadap *meta-learner* pada *Super Learner* meliputi dua metode yang berbeda, yakni NNloglik dan AUC-maxim [10].

Di samping itu, untuk mempercepat waktu komputasi, pembangunan model prediksi ini mengimplementasikan komputasi paralel dalam lingkungan komputasi berkinerja tinggi (*high-performance computing*/HPC). Penelitian ini pun bereksperimen untuk menemukan strategi paralelisasi yang paling efisien. Eksperimen dilakukan dengan membandingkan komputasi paralel menggunakan jenis prosesor yang berbeda-beda pada mesin HPC.

Dengan demikian, penelitian ini ditujukan untuk membangun suatu model prediksi yang mampu mendeteksi *dropout* pada MOOC secara akurat, dan disertai dengan adanya efisiensi waktu pembangunan model. Hasil prediksi *dropout* diharapkan dapat membantu pihak penyedia platform MOOC dalam memberikan intervensi pencegahan *dropout* sedini mungkin guna meningkatkan retensi siswa sekaligus membuat siswa MOOC meraih pengalaman pembelajaran yang lebih berkualitas.



Gambar 1. Cara kerja *super learner*.

II. TINJAUAN PUSTAKA

A. *Stacking Super Learner*

Super learner merupakan salah satu algoritme dari metode ansambel *stacking* yang mengombinasikan hasil prediksi dari sejumlah pembelajar dasar melalui adanya *meta-learner*. Pembelajar dasar merupakan kumpulan pengklasifikasi (*classifier*) dengan algoritme berbeda-beda yang mana hasil prediksinya akan dikombinasikan. *Meta-learner*, yaitu algoritme yang berperan dalam mengombinasikan hasil prediksi dari masing-masing pembelajar dasar [11–12]. Pada tahun 2007, algoritme *Super Learner* diciptakan sebagai perbaikan dari algoritme *Stacked Generalization*, yang mana *meta-learner* pada *Super Learner* secara otomatis akan mengombinasikan algoritme-algoritme pembelajar dasar dengan bobot kombinasi yang optimal. Definisi "optimal" yang dimaksud, yaitu dicapainya nilai optimum pada fungsi obyektif tertentu, yang dapat berupa maksimasi ataupun minimasi terhadap fungsi kerugian [13].

Cara kerja *Super Learner* ditunjukkan pada Gambar 1. Pertama-tama, data latih dibagi menjadi V-*fold* untuk kemudian dilakukan pelatihan model pembelajar dasar secara validasi silang (*cross-validation*). Saat melakukan tahapan tersebut, hasil prediksi dari setiap pembelajar dasar terhadap set data validasi di setiap *fold* perlu disimpan dan disatukan ke dalam suatu matriks *Z*. Matriks ini digunakan untuk melatih algoritme *meta-learner* agar mampu meminimumkan *cross-validated risk*, yaitu besarnya risiko (berdasarkan fungsi kerugian) dari hasil prediksi set data validasi yang dijelaskan sebelumnya. Jika risiko (R_{CV}) telah minimal, maka dapat dikatakan bahwa bobot-bobot pembelajar dasar (β) yang diberikan oleh *meta-learner* sudah optimal seperti pada Persamaan (1).

$$\hat{\beta} = \arg min_{\beta}\, R_{CV}(\beta) \qquad (1)$$

Terakhir, evaluasi *Super Learner* dilakukan dengan melatih kembali masing-masing pembelajar dasar terhadap keseluruhan data latih, lalu pembelajar dasar perlu memprediksi data latih ataupun data uji, dan prediksi-prediksi tersebut dikombinasikan menggunakan *meta-learner* yang telah dioptimalkan [9],13].

B. *Meta-learner* NNloglik

NNloglik merupakan *meta-learner* yang menggunakan algoritme LogReg dengan disertai adanya *non-negative binomial likelihood maximization* yang merupakan bentuk penerapan metode MLE (*maximum likelihood estimation* [14]. *Likelihood* merupakan fungsi dari parameter yang menunjukkan seberapa mungkin data observasi (aktual) akan terjadi jika menggunakan parameter estimasi di suatu model statistik [15]. MLE ini ekuivalen dengan meminimumkan *residual deviance*-nya. Persamaan *residual deviance* untuk model binomial, yaitu (2) dan persamaan ini dijadikan sebagai fungsi kerugian pada metode NNloglik. Notasi p adalah prediksi terhadap label sebenarnya (y) yang bernilai 0 atau 1 [16].

$$D = -2 \sum_{i=1}^{n} y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i), y_i \in \{0,1\} \qquad (2)$$

Sementara itu, LogReg sebagai algoritme *meta-learner* ini mengombinasikan pembelajar dasar melalui fungsi logistik sehingga dihasilkan prediksi *Super Learner* (p') yang berupa probabilitas dengan rentang nilai $[0,1]$ [17]. Persamaan LogReg pada NNloglik ditunjukkan pada Persamaan (3) [18].

$$p' = \frac{1}{1 + e^{-(\boldsymbol{\beta_1} x_1 + \beta_2 x_2 + \cdots + \beta_k x_k)}}, \beta_k \geq 0 \; \forall \, k, \sum_{k=1}^{K} \beta_k = 1 \qquad (3)$$

Notasi $\beta_k$menunjukkan koefisien/bobot untuk pembelajar dasar ke- k dan x_{ik} adalah hasil prediksi pembelajar dasar ke-k pada baris data ke- i yang ada dalam matriks *Z*. Intersep,

	enroll_id	username	course_id	session_id	action	time
0	772	5981	4095	d8a9b787fa69063c34c73b9c29190b1c	click_about	2015-09-27T15:42:59
1	772	5981	4095	d8a9b787fa69063c34c73b9c29190b1c	click_info	2015-09-27T15:43:12
2	773	1544995	4095	2f02b86eb3ea2cbf0be11385a8dc62e5	pause_video	2015-10-19T19:37:42
3	773	1544995	4095	2f02b86eb3ea2cbf0be11385a8dc62e5	load_video	2015-10-19T19:33:27
4	773	1544995	4095	2f02b86eb3ea2cbf0be11385a8dc62e5	play_video	2015-10-19T19:33:30
...	...	...	...	...	...	...
42110397	466785	2513464	4227	b12bc83e13cd943cf61bf78f09c72158	problem_check_incorrect	2016-03-19T21:37:15
42110398	466785	2513464	4227	b12bc83e13cd943cf61bf78f09c72158	problem_check_incorrect	2016-03-19T21:37:22
42110399	466785	2513464	4227	b12bc83e13cd943cf61bf78f09c72158	problem_check_correct	2016-03-19T21:37:37
42110400	466785	2513464	4227	7eca0904ae14dc8af809c0362632dd8e	click_courseware	2016-03-19T19:24:44
42110401	466785	2513464	4227	b12bc83e13cd943cf61bf78f09c72158	problem_get	2016-03-19T21:38:34

42110402 rows × 6 columns

(a)

	enroll_id	username	course_id	course_type	course_category	course_start_date	course_end_date	dropout
106665	772	5981	4095	instructor-paced	art	2015-09-25 08:00:00	2016-01-06 08:00:00	1
156597	773	1544995	4095	instructor-paced	art	2015-09-25 08:00:00	2016-01-06 08:00:00	1
88668	774	1072798	4095	instructor-paced	art	2015-09-25 08:00:00	2016-01-06 08:00:00	1
209389	775	1520977	4095	instructor-paced	art	2015-09-25 08:00:00	2016-01-06 08:00:00	1
204021	776	561867	4095	instructor-paced	art	2015-09-25 08:00:00	2016-01-06 08:00:00	0
...	...	...	...	...	...	...	...	...
167246	466781	2830711	4227	instructor-paced	chemistry	2016-03-01 08:00:00	2016-06-29 00:00:00	1
223000	466782	2680742	4227	instructor-paced	chemistry	2016-03-01 08:00:00	2016-06-29 00:00:00	1
159583	466783	2665176	4227	instructor-paced	chemistry	2016-03-01 08:00:00	2016-06-29 00:00:00	1
80287	466785	2513464	4227	instructor-paced	chemistry	2016-03-01 08:00:00	2016-06-29 00:00:00	1
52670	466786	2659552	4227	instructor-paced	chemistry	2016-03-01 08:00:00	2016-06-29 00:00:00	0

225642 rows × 8 columns

(b)

Gambar 2. Pratinjau data: (a) *Clickstream* dan (b) data informasi siswa.

β_0, tidak digunakan dalam NNloglik. Hal tersebut dikarenakan *Super Learner* berupaya untuk mencari kombinasi terbobot yang terbaik dari k pembelajar dasar sehingga diharapkan koefisien $\{\beta_1, \dots, \beta_k\}$ dapat seoptimal mungkin, dengan mengabaikan intersep [19]. Selain itu, koefisien β_k pada NNloglik memiliki syarat (*constraint*), yaitu koefisien harus bernilai positif dan nilai koefisien perlu dinormalisasi sedemikian rupa sehingga total koefisien sama dengan satu [10].

C. *Meta-learner* AUC-maxim

AUC-maxim merupakan *meta-learner* yang menggunakan algoritme *Linear Regression* (LinReg) dengan disertai adanya maksimasi skor AUC, yakni metrik yang mengukur luas di bawah kurva ROC (*receiver operating characteristic*). Semakin nilai AUC mendekati 1, maka semakin baik model tersebut dalam membedakan kelas positif dari kelas negatif. Maksimasi AUC ini ekuivalen dengan meminimumkan *rank loss* seperti pada (4) dan persamaan ini dijadikan sebagai fungsi kerugian pada metode AUC-maxim.

$$rank\ loss = 1 - AUC \tag{4}$$

LinReg sebagai algoritme *meta-learner*, seperti pada (5), mengombinasikan hasil prediksi pembelajar dasar secara linier sebagai prediksi dari *Super Learner* (p'). Pada AUC-maxim, persamaan LinReg yang digunakan juga mengabaikan intersep (β_0) untuk mengoptimalkan koefisien $\{\beta_1, \dots, \beta_k\}$ yang berlaku sebagai bobot pembelajar dasar ke-k. Koefisien β_k tersebut juga perlu memenuhi syarat, yakni bernilai positif dan total koefisien harus sama dengan satu sehingga dilakukan normalisasi koefisien jika perlu [10],[14].

$$p' = \beta_1 x_1 + \cdots + \beta_k x_k, \beta_k \geq 0 \ \forall\, k, \sum_{k=1}^{K} \beta_k = 1 \tag{5}$$

D. *Algoritme untuk Pembelajar Dasar*

Algoritme pembelajar dasar yang digunakan untuk membangun model *Super Learner* meliputi LogReg, SVM, KNN, NB, RF, dan XGBoost. LogReg merupakan model statistik untuk klasifikasi yang berbasis pada fungsi logistik [20]. SVM bekerja dengan menggunakan *kernel* untuk memproyeksikan variabel *input* ke dalam ruang *n* dimensi, kemudian klasifikasi dilakukan dengan membagi ruang menjadi dua area terpisah untuk masing-masing kelas [21]. KNN mengasumsikan bahwa elemen yang serupa (*similar*) akan saling berdekatan (*close proximity*) sehingga KNN akan mengklasifikasikan sebuah titik x dengan melihat kelas yang dominan dari k titik sampel terdekatnya (*neighbours*) [20],[22]. NB adalah pengklasifikasi probabilistik berdasarkan teorema Bayes dengan asumsi yang kuat mengenai indepen-densi antar fitur [23]. RF merupakan metode ansambel dalam kategori *bagging* yang melatih sekumpulan DT secara paralel dan kemudian hasil akhir klasifikasi adalah kelas yang mayoritas dari hasil prediksi kumpulan DT tersebut [20],[24]. XGBoost merupakan metode ansambel dalam kategori *boosting* yang mengombinasikan prediksi dari sejumlah DT melalui strategi pelatihan model yang aditif. Maksud dari pelatihan model secara aditif, yaitu di tiap iterasinya, sebuah DT akan dilatih berdasarkan residual dari DT sebelumnya agar bias semakin berkurang, kemudian pada akhirnya, hasil prediksi dari seluruh DT akan dijumlahkan [25].

Tabel 1.
Bobot Pembelajar Dasar di Setiap Model *Super Learner*

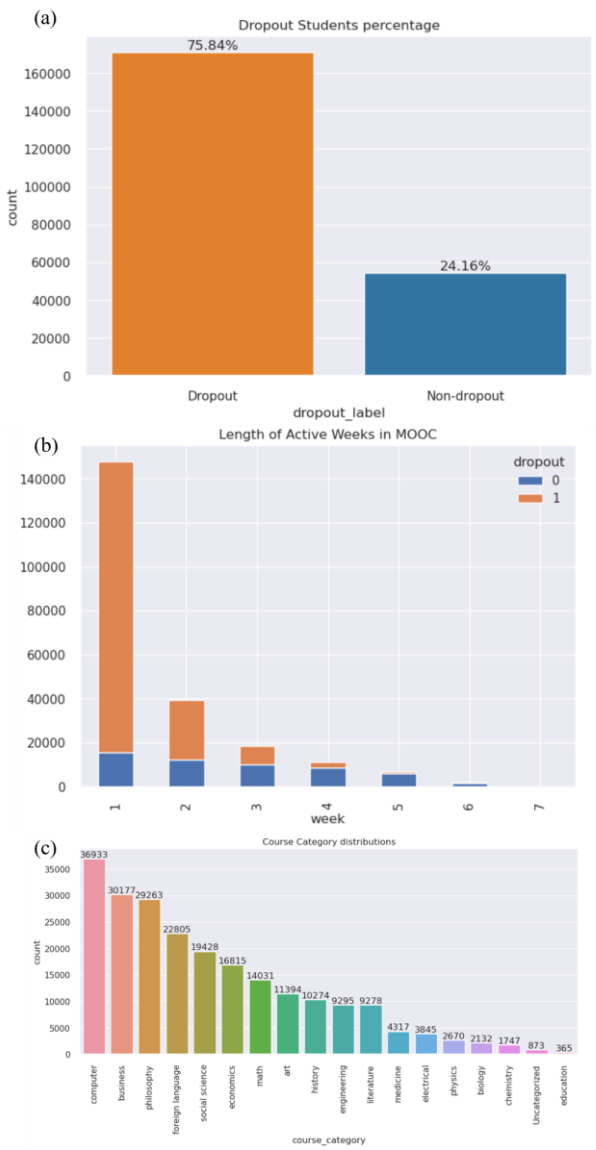
Pembelajar Dasar	Bobot Pembelajar Dasar pada SL-NNloglik	Bobot Pembelajar Dasar pada SL-AUC-maxim
SVM	0	0
LogReg	0,019	0
KNN	0,007	0,02
NB	0	0
RF	0,148	0,15
XGBoost	0,826	0,83
Total Bobot	1	1

Tabel 2.
Evaluasi Risiko Model Pembelajar Dasar dan *Super Learner*

Model Pembelajar Dasar atau *Super Learner* (SL)	Risiko	
	Binomial Residual Deviance	*Rank Loss*
SVM	0,94	0,37
LogReg	1,13	0,28
KNN	1,58	0,29
NB	2,3	0,38
RF	0,78	0,27
XGBoost	0,76	0,27
SL-NNloglik	0,72	0,25
SL-AUC-maxim	0,72	0,25

E. *Paralelisasi dalam Lingkungan HPC*

Komputasi berkinerja tinggi (HPC) ditujukan untuk menyelesaikan operasi perhitungan yang kompleks dalam waktu singkat. Mesin HPC ditenagai oleh banyak prosesor yang membuatnya mampu melakukan berbagai kalkulasi secara bersamaan atau secara paralel. Komponen utama HPC adalah prosesor berupa GPU dan CPU. *Graphic Processing Unit* (GPU) memiliki banyak *computing core* sehingga menjadikan GPU mampu menangani komputasi data jumlah besar sekaligus secara bersamaan (*high throughput*) yang berdampak pada berkurangnya latensi. GPU melakukan paralelisme yang berbasis data (*data-based parallelism*), yakni membagi data menjadi beberapa bagian dan mendistribusikannya kepada masing-masing *computing core*, kemudian seluruh *computing core* akan secara bersama-sama mengeksekusi *task* yang sama. Di sisi lain, *Central Processing Unit* (CPU) mampu menjalankan sebuah instruksi dengan sangat cepat (*low latency*), yang mana tiap instruksinya dieksekusi secara



Gambar 3. Analisis data eksploratif: (a) Proporsi kelas *dropout* dan non-*dropout*, (b) panjang minggu aktif siswa di MOOC berdasarkan kelas *dropout*/non-*dropout*, dan (c) distribusi bidang kursus.

sekuensial sehingga hal ini berguna bagi *task* yang memerlukan dependensi antar data. Pada *multi-cores* CPU, apabila seluruh *core* digunakan bersamaan, maka CPU melakukan paralelisme berbasis *task* (*task-based parallelism*), yakni memproses sejumlah *task* secara bersamaan dengan mendistribusikan *task* kepada masing-masing *core* [26].

III. METODOLOGI

A. Pengumpulan Data

Dataset yang digunakan adalah data *clickstream* dari siswa MOOC XuetangX pada 247 kursus daring yang berlangsung di tahun 2015–2017. Data ini merupakan dataset yang digunakan oleh Feng *et al.* (2019) [27]. Dataset tersebut dapat diunduh oleh publik melalui tautan http://moocdata.cn. Seperti pada Gambar 2, dataset ini terbagi menjadi data *clickstream* (Gambar 2(a)) dan data informasi siswa MOOC (Gambar 2(b)) yang menunjukkan apakah seorang siswa pada akhirnya *dropout* atau tidak. Pada dataset ini, sampel *clickstream* di setiap kursus daring rata-rata diambil dalam kurun waktu 36 hari (6–7 minggu) sejak kursus tersebut dimulai, namun terdapat beberapa kursus daring yang sampelnya diambil dalam kurun waktu < 36 hari.

Tabel 3.
Perbandingan Kinerja Antarmodel

Peringkat	Model	Skor F1
1	SL-AUC-maxim	0,90147
2	SL-NNloglik	0,90126
3	XGBoost	0,90123
4	SG-LogReg	0,90119
5	SG-XGBoost	0,90002
6	SG-SVM	0,90000
7	SG-RF	0,89902
8	RF	0,89839
9	KNN	0,89023
10	SG-KNN	0,88880
11	SG-NB	0,87946
12	SVM	0,87440
13	NB	0,85903
14	LogReg	0,83983

B. Analisis Data Eksploratif

Proporsi siswa yang *dropout*, yaitu 75,8% dan siswa *non-dropout* sebesar 24,16% seperti pada Gambar 3(a). Meskipun distribusi kelas pada dataset ini tidak seimbang (*imbalanced*), namun pembangunan model ini difokuskan untuk memprediksi kelas *dropout* yang merupakan kelas mayoritas sehingga ketidakseimbangan ini tidak menjadi suatu masalah yang berarti. Pada Gambar 3(b), ditunjukkan bahwa sebagian besar siswa *dropout* telah berhenti mempelajari kursusnya sejak minggu pertama. Di sisi lain, panjang minggu aktif siswa non-*dropout* cenderung terdistribusi merata, yakni terdapat siswa non-*dropout* yang menyelesaikan kursusnya hanya dalam satu minggu, dua minggu, dan seterusnya hingga lima minggu. Di sisi lain, siswa dengan panjang minggu aktif mencapai enam atau tujuh minggu relatif sangat jarang.

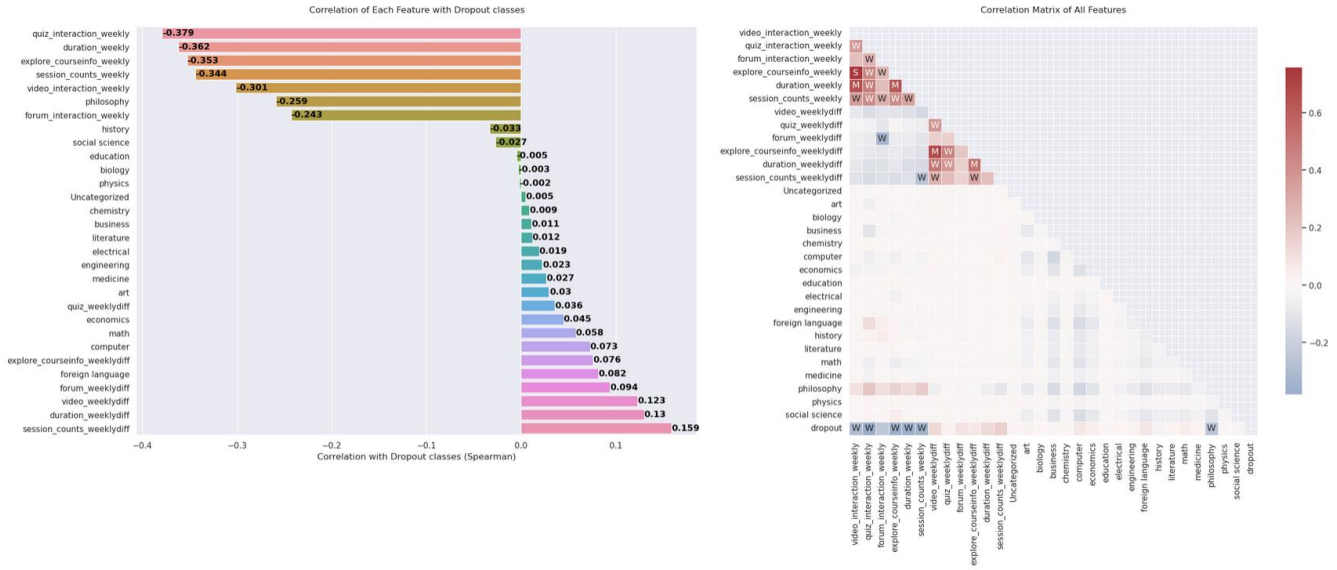
Kemudian, jika dikategorikan berdasarkan bidangnya seperti Gambar 3(c), maka lima bidang kursus yang paling diminati adalah komputer, bisnis, filosofi, bahasa asing, dan ilmu sosial dengan masing-masing siswanya mencapai $\pm$20.000–35.000 orang. Sementara itu, pada posisi dua terbawah, terdapat bidang pendidikan dan bidang tanpa kategori (*unrecognized*) yang masing-masing memiliki siswa < 1.000 orang.

C. Praproses Data

Tahap praproses data meliputi rekayasa fitur, seleksi fitur, pembagian data latih dan data uji, penanganan data pencilan, dan penskalaan fitur. Rekayasa fitur dilakukan dengan terlebih dahulu melakukan *one-hot encoding* terhadap kolom "action" dan menghitung durasi sesi *login* berdasarkan informasi pada kolom "time", yang mana kolom-kolom tersebut terdapat pada tabel dalam Gambar 2(a). Kemudian, menjumlahkan frekuensi *clickstream* siswa terhadap video materi, kuis, forum diskusi, dan laman informasi kursus, serta total durasi dan jumlah sesi *login* di setiap minggunya. Selanjutnya, data siswa pada keseluruhan minggu dirata-rata secara terbobot berdasarkan keterkinian minggunya sehingga tiap siswa kini hanya memiliki satu baris data. Pembobotan data tersebut mengikuti fungsi eksponensial seperti (6), dengan $\frac{1}{5}$ sebagai parameter yang mengatur seberapa cepat bobot akan mengecil untuk data lampau ke-t ($t = 0,1,2,\dots,t$), yang mana bobot untuk data minggu terakhir adalah $\text{bobot}(0) = 1$ [28].

$$\text{bobot}(t) = e^{-\frac{1}{5}\cdot t} \quad (6)$$

Lalu, kolom "course_category" juga digunakan sebagai fitur. Namun, karena tipe datanya kategorial, maka dilakukan



Gambar 4. Korelasi fitur dengan target dan korelasi antar fitur.

pengkodean kategorial menggunakan *one-hot encoding* sehingga menghasilkan 18 kolom baru untuk masing-masing bidang kursus (*course category*).

Seleksi fitur kemudian dilakukan dengan melihat kekuatan korelasi *Spearman* antara fitur dengan target (kolom "*dropout*") dan korelasi terhadap fitur lainnya seperti pada Gambar 4. Fitur yang terkait dengan *clickstream* video akhirnya tidak digunakan karena berkorelasi kuat dengan fitur *clickstream* laman informasi kursus. Hal ini mengindikasikan bahwa fitur tersebut redundan. Dalam hal ini, fitur *clickstream* laman informasi kursus lebih dipilih karena korelasinya terhadap target lebih kuat.

Setelah seleksi fitur, maka data selanjutnya dibagi menjadi 70% untuk data latih dan 30% sebagai data uji. Kemudian, pada data latih, dilakukan penanganan data pencilan menggunakan teknik winsorisasi standar deviasi. Terakhir, melakukan penskalaan fitur dengan teknik normalisasi (MinMaxScaler) terhadap data latih maupun data uji. Penskalaan fitur ini diperlukan khususnya untuk algoritme pembelajar dasar berbasis jarak seperti SVM dan KNN yang sensitif terhadap perbedaan skala fitur dan juga dapat mempercepat konvergensi pada pembelajar dasar yang berbasis gradien seperti LogReg [29–30].

D. Pembangunan Model Prediksi dengan Super Learner

Eksperimen pembangunan model ini dilakukan dengan memanfaatkan komputasi paralel pada GPU ataupun CPU sehingga kecepatan komputasi antarprosesor dapat dibandingkan. Model pembelajar dasar XGBoost dibangun menggunakan *library* xgboost, sedangkan pembelajar dasar lainnya menggunakan *library* cuML jika diimplementasikan pada GPU dan *library* scikit-learn apabila pada CPU.

*Hyperparame*ter *tuning* pun dilakukan terhadap setiap model pembelajar dasar dengan teknik *grid search* yang menggunakan validasi silang 5-*fold*. Setelah diperoleh model dengan set *hyperparameter* terbaik, maka model pembelajar dasar perlu divalidasi silang (5-*fold*) kembali terhadap data latih untuk memprediksi set data validasi pada setiap *fold*-nya. Hasil prediksi dari seluruh pembelajar tersebut kemudian disatukan ke dalam matriks Z yang digunakan untuk melatih *meta-learner*.

Meta-learner dioptimasi menggunakan matriks Z dan label aktualnya (*y*) pada data latih. *Meta-learner* NNloglik dioptimasi untuk meminimumkan *residual deviance*, sedangkan AUC-maxim berupaya meminimumkan *rank loss*. Akhirnya, diperoleh bobot pembelajar dasar yang optimal $(\hat{\beta})$ seperti pada Tabel 1. Dari Tabel 1, terlihat bahwa XGBoost merupakan pembelajar dasar dengan bobot yang paling signifikan sehingga dapat dikatakan bahwa prediksi model XGBoost adalah yang paling menentukan prediksi *Super Learner* nantinya. Sebaliknya, SVM dan NB secara konsisten berbobot 0 pada *meta-learner* NNloglik maupun AUC-maxim yang artinya hasil prediksi kedua model tersebut tidak ikut dikombinasikan dalam *Super Learner*.

Terakhir, evaluasi hasil prediksi *Super Learner* dilakukan dengan melatih ulang pembelajar dasar terhadap keseluruhan data latih. Lalu, pembelajar dasar memprediksi data latih ataupun data uji, *Super Learner* kemudian mengombinasikan hasil prediksi dari seluruh pembelajar dasar menggunakan *meta-learner* yang telah dioptimasi.

E. Uji Coba dan Evaluasi Model

Risiko (*risk*) antara model pembelajar dasar dengan model *Super Learner* pun perlu dibandingkan untuk mengevaluasi apakah model *Super Learner* berhasil mengombinasikan pembelajar dasar dengan optimal sehingga dicapai risiko yang lebih minimum daripada risiko pembelajar dasar. Evaluasi risiko ini dapat dilihat pada Tabel 2 yang menunjukkan bahwa *Super Learner* memiliki risiko *residual deviance* dan *rank loss* yang paling minimum. Baik *Super Learner* (SL) dengan *meta-learner* NNloglik maupun AUC-maxim memiliki nilai *residual deviance* terendah sebesar 0,72 dan *rank loss* terendah sebesar 0,25. Dengan begitu, dapat dikatakan bahwa model *Super Learner* yang dihasilkan telah mengombinasikan pembelajar dasar secara optimal sehingga dicapai risiko yang minimum.

Selanjutnya, seperti pada Tabel 3, kinerja model *Super Learner* (SL) dalam memprediksi data uji dibandingkan dengan kinerja model lainnya, yaitu tiap-tiap pembelajar dasar dan *Stacked Generalization* (SG) dengan *meta-learner* yang berbeda-beda sebagai algoritme *stacking* pembanding. Secara lebih spesifik, model SG yang dijadikan sebagai bahan perbandingan meliputi SG yang menggunakan *meta-learner* LogReg (SG-LogReg), SG-KNN, SG-NB, SG-RF, dan SG-XGBoost. Metrik yang digunakan, yaitu skor F1 (F1 *score*).

Tabel 4.
Perbandingan Waktu Eksekusi Antar Prosesor

Kode Program*	Jenis Prosesor**				Perbandingan GPU dengan CPU Tercepat
	1 GPU RAM 20 GB	1 GPU RAM 40 GB	1 CPU RAM 60 GB, 12 *core*	1 CPU RAM 120 GB, 28 *core*	
Hyperpara-meter tuning (*grid search*, 5-*fold*)	6 menit 37 detik	7 menit 6 detik	24 menit 23 detik	15 menit 35 detik	2,4×
Cross-validation training (5-*fold*)	55 detik	1 menit 8 detik	4 menit 36 detik	2 menit 38 detik	2,9×
Training dengan seluruh data latih	12 detik	13 detik	1 menit 9 detik	45 detik	3,75×
Inference terhadap data latih	1,4 detik	1,2 detik	28 detik	31 detik	23,3×
Inference terhadap data uji	0,4 detik	0,4 detik	7 detik	10 detik	17,5×

*) masing-masing kode program dijalankan terhadap semua pembelajar dasar
**) prosesor di atas merupakan prosesor yang terdapat dalam mesin NVIDIA DGX A100

Pada peringkat 1 dan 2, terdapat model *Super Learner* dengan *meta-learner* AUC-maxim dan NNloglik yang secara berurutan mencapai skor F1 sebesar 0,90147 dan 0,90126. Sementara itu, XGBoost berada pada peringkat 3 dan menjadi model pembelajar dasar terbaik dengan skor F1 sebesar 0,90123. Kemudian, ditemukan bahwa model *Stacked Generalization* justru berperforma lebih rendah daripada ketika hanya menggunakan pembelajar dasar XGBoost. Model SG-LogReg sebagai model *Stacked Generalization* terbaik hanya mencapai skor F1 sebesar 0,90119. Dengan demikian, ditemukan bahwa model dengan skor F1 yang paling unggul adalah model *Super Learner*, baik yang menggunakan *meta-learner* AUC-maxim maupun NNloglik, daripada model-model *Stacked Generalization* ataupun masing-masing model pembelajar dasar (tanpa *stacking*).

Selain kinerja model, waktu komputasi pun dianalisis, yakni ketika dilakukan komputasi yang memanfaatkan paralelisasi prosesor GPU ataupun prosesor CPU dalam lingkungan HPC. Mesin HPC yang digunakan, yaitu NVIDIA DGX A100. Pada HPC tersebut, dilakukan eksperimen paralelisasi terhadap beberapa jenis prosesor seperti pada Tabel 4. Adapun paralelisasi yang disoroti dalam hal ini, yaitu paralelisasi terhadap eksekusi algoritme pembelajaran mesin. Hasil yang diperoleh menunjukkan bahwa urutan prosesor dari yang paling cepat, yaitu (i) satu GPU dengan RAM 20 GB, (ii) satu GPU dengan RAM 40 GB, (iii) satu CPU 28 *core* dengan total RAM 120 GB, dan (iv) satu CPU 12 *core* dengan total RAM 60 GB. Dari hasil tersebut, diketahui bahwa eksekusi kode program pada GPU bisa 2,4–23,3 kali lebih cepat daripada CPU. Hal ini mengindikasikan bahwa paralelisasi pada GPU dalam hal ini dinilai lebih efisien daripada paralelisasi terhadap CPU. Selain itu, ditemukan bahwa GPU dengan ukuran RAM yang lebih besar justru menjadi sedikit lebih lambat dalam percobaan ini. Hal tersebut dapat terjadi karena GPU yang mengalami *under-utilization.* Permasalahan ini muncul ketika ukuran data terlalu kecil dan/atau skala paralelisasi yang dilakukan terlalu kecil sehingga biaya (*cost*) untuk transmisi data dari memori ke GPU menjadi lebih mahal daripada keuntungan yang diperoleh dari proses paralelisasi tersebut. Apabila terjadi *underutilization* pada GPU, maka waktu eksekusi akan menjadi lebih lambat dari semestinya [31]. Oleh sebab itu, penggunaan GPU dengan RAM 20 GB dinilai lebih cocok untuk percobaan ini daripada RAM 40 GB, mengingat data latih maupun data uji terdiri atas < 200.000 baris data serta berdimensi < 50 fitur. Lain halnya pada CPU di percobaan ini yang mengalami peningkatan performa seiring dengan bertambahnya ukuran memori dan jumlah *core*, meskipun percepatan (*speedup*) yang dihasilkan belum sebaik GPU.

IV. KESIMPULAN

Dari hasil penelitian ini, didapatkan bahwa model terbaik adalah model *Super Learner* dengan *meta-learner* AUC-maxim mencapai skor F1 tertinggi, yaitu 0,90147 dan disusul oleh model *Super Learner* dengan *meta-learner* NNloglik yang mencapai skor F1 sebesar 0,90126. Sementara itu, di antara pembelajar dasar SVM, LogReg, KNN, NB, RF, dan XGBoost, didapatkan XGBoost sebagai model pembelajar dasar terbaik yang hanya mencapai skor F1 sebesar 0,90123. Kemudian, jika dibandingkan dengan model-model *Stacked Generalization* (SG) yang menggunakan *meta-learner* Log-Reg (SG-LogReg), SG-KNN, SG-NB, SG-RF, dan SG-XGBoost, ternyata diperoleh model SG-LogReg sebagai model *Stacked Generalization* terbaik yang bahkan mencapai skor F1 lebih rendah, yaitu 0,90119. Dengan demikian, kedua model *Super Learner* berhasil mengungguli kinerja dari model pembelajar dasar yang menyusunnya maupun model yang menggunakan algoritme *Stacked Generalization*.

Paralelisasi komputasi juga telah dilakukan dengan mesin HPC NVIDIA DGX A100. Hasil eksperimen menunjukkan bahwa urutan prosesor dari yang paling cepat, yaitu (i) satu GPU dengan RAM 20 GB, (ii) satu GPU dengan RAM 40 GB, (iii) satu CPU 28 *core* dengan total RAM 120 GB, dan (iv) satu CPU 12 *core* dengan total RAM 60 GB. Pada percobaan ini, GPU dengan RAM yang lebih besar justru mengalami *underutilization* sehingga membuat waktu eksekusi menjadi sedikit lebih lambat. Di sisi lain, CPU justru mengalami peningkatan performa seiring dengan bertambahnya ukuran memori dan jumlah *core*. Meskipun demikian, paralelisasi pada GPU nyatanya 2,4–23,3 kali lebih cepat daripada paralelisasi CPU dalam percobaan ini.

DAFTAR PUSTAKA

[1] K. F. Hew and W. S. Cheung, "Students' and instructors' use of massive open online courses (MOOCs): Motivations and challenges," *Educ. Res. Rev.*, vol. 12, pp. 45–58, 2014, doi: 10.1016/j.edurev.2014.05.001.

[2] M. H. Baturay, "An Overview of the World of MOOCs," in *Procedia - Social and Behavioral Sciences*, 2015, vol. 174, pp. 427–433. doi: 10.1016/j.sbspro.2015.01.685.

[3] Y. Goel and R. Goyal, "On the effectiveness of self-training in mooc dropout prediction," *Open Comput. Sci.*, vol. 10, no. 1, pp. 246–258, 2020, doi: 10.1515/comp-2020-0153.

[4] A. A. AlDahdouh and A. J. Osório, "Planning to design MOOC? think first!," *Online J. Distance Educ. e-Learning*, vol. 4, no. 2, pp. 47–57, Apr. 2016, doi: 10.5281/zenodo.48804.

[5] J. Gardner and C. Brooks, "Student success prediction in MOOCs," *User Model. User-adapt. Interact.*, vol. 28, no. 2, pp. 127–203, 2018,

doi: 10.1007/s11257-018-9203-z.
[6] W. Xing, X. Chen, J. Stein, and M. Marcinkowski, "Temporal predication of dropouts in MOOCs: Reaching the low hanging fruit through stacking generalization," *Comput. Human Behav.*, vol. 58, pp. 119–129, 2016, doi: 10.1016/j.chb.2015.12.007.
[7] M. Youssef, S. Mohammed, E. K. Hamada, and B. F. Wafaa, "A predictive approach based on efficient feature selection and learning algorithms' competition: case of learners' dropout in MOOCs," *Educ. Inf. Technol.*, vol. 24, no. 6, pp. 3591–3618, 2019, doi: 10.1007/s10639-019-09934-y.
[8] S. Lankford and D. Grimes, "Enhanced Neural Architecture Search Using Super Learner and Ensemble Approaches," in *2nd Asia Service Sciences and Software Engineering Conference*, New York, 2021, pp. 137–143. doi: 10.1145/3456126.3456133.
[9] G. Li, M. Shen, M. Li, and J. Cheng, "Personal credit default discrimination model based on super learner ensemble," *Math. Probl. Eng.*, vol. 2021, p. 5586120, 2021, doi: 10.1155/2021/5586120.
[10] E. Polley and M. J. van der Laan, "SuperLearner: Super Learner Prediction, 2013," *Comprehensive R Archive Network*, 2019. Tersedia pada: https://cran.r-project.org/web/packages/SuperLearner/index.html
[11] A. I. Naimi and L. B. Balzer, "Stacked generalization: an introduction to super learning," *Eur. J. Epidemiol.*, vol. 33, no. 5, pp. 459–464, May 2018, doi: 10.1007/s10654-018-0390-z.
[12] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992, doi: https://doi.org/10.1016/S0893-6080(05)80023-1.
[13] M. J. Van Der Laan, E. C. Polley, and A. E. Hubbard, "Super learner," *Stat. Appl. Genet. Mol. Biol.*, vol. 6, no. 1, 2007, doi: 10.2202/1544-6115.1309.
[14] E. LeDell, M. J. Van Der Laan, and M. Peterson, "AUC-maximizing ensembles through metalearning," *Int. J. Biostat.*, vol. 12, no. 1, pp. 203–218, 2016, doi: 10.1515/ijb-2015-0035.
[15] M. D. Edge, *Statistical Thinking from Scratch: A Primer for Scientists*. Oxford: Oxford University Press, 2017. ISBN: 9780198827627.
[16] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*, 5th ed. New Jersey: John Wiley & Sons. ISBN: 9780470542811.
[17] V. Nasteski, "An overview of the supervised machine learning methods," *Horizons*, vol. 4, pp. 51–62, 2017, doi: 10.20544/horizons.b.04.1.17.p05.
[18] J. M. Hilbe, *Practical Guide to Logistic Regression*, 1st ed. Boca Raton: Chapman & Hall CRC, 2015. ISBN: 978-1498709576.
[19] S. Zian, S. A. Kareem, and K. D. Varathan, "An empirical evaluation of stacked ensembles with different meta-learners in imbalanced classification," *IEEE Access*, vol. 9, pp. 87434–87452, 2021, doi: 10.1109/ACCESS.2021.3088414.
[20] A. C. Lorena *et al.*, "Comparing machine learning classifiers in potential distribution modelling," *Expert Syst. Appl.*, vol. 38, no. 5, pp. 5268–5275, 2011, doi: https://doi.org/10.1016/j.eswa.2010.10.031.
[21] A. Abdiansah and R. Wardoyo, "Time complexity analysis of support vector machines (svm) in libsvm," *Int. J. Comput. Appl.*, vol. 128, no. 3, pp. 28–34, 2015, doi: 10.5120/ijca2015906480.
[22] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *Am. Stat.*, vol. 46, no. 3, p. 175, 1992, doi: 10.2307/2685209.
[23] K. Vembandasamy, R. Sasipriya, and E. Deepa, "Heart diseases detection using naive bayes algorithm," *Int. J. Innov. Sci. Eng. Technol.*, vol. 2, no. 3, pp. 441–444, 2015.
[24] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
[25] J. Fan *et al.*, "Evaluation of svm, elm and four tree-based ensemble models for predicting daily reference evapotranspiration using limited meteorological data in different climates of China," *Agric. For. Meteorol.*, vol. 263, pp. 225–241, 2018, doi: 10.1016/j.agrformet.2018.08.019.
[26] S. Cook, *CUDA Programming: A Developer's Guide to Parallel Computing with GPUs*, 1st ed. Waltham: Morgan Kaufmann, 2012. ISBN: 978-0-12-415933-4.
[27] W. Feng, J. Tang, and T. X. Liu, "Understanding Dropouts in MOOCs," in *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, Honolulu, 2019, pp. 517–524. doi: 10.1609/aaai.v33i01.3301517.
[28] Y. Ding and X. Li, "Time Weight Collaborative Filtering," in *International Conference on Information and Knowledge Management*, New York, 2005, pp. 485–492. doi: 10.1145/1099554.1099689.
[29] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Appl. Soft Comput.*, vol. 97, p. 105524, 2020, doi: 10.1016/j.asoc.2019.105524.
[30] S. Bharadwaj, B. S. Anil, A. Pahargarh, A. Pahargarh, P. S. Gowra, and S. Kumar, "Customer Churn Prediction in Mobile Networks using Logistic Regression and Multilayer Perceptron(MLP)," in *Proceedings of the 2nd International Conference on Green Computing and Internet of Things*, Bangalore, 2018, pp. 436–438. doi: 10.1109/ICGCIoT.2018.8752982.
[31] M. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, and J. Kautz, "Reinforcement Learning through Asynchronous Advantage Actor-Critic on a GPU," in *International Conference on Learning Representations*, Bangalore, 2016, pp. 1–12, doi: https://doi.org/10.48550/arXiv.1611.06256.