XGBOOST DENGAN *RANDOM SEARCH HYPER-PARAMETER TUNING* UNTUK KLASIFIKASI SITUS *PHISING*

Muhammad Ryan Afrizal[a1], Muliadi[a2], Radityo Adi Nugroho[a3], Dwi Kartini[a4], Rudy Herteno[a5]

[a]Ilmu Komputer Fakultas Matematika dan Ilmu Alam Universitas Lambung Mangkurat
Jalan Ahmad Yani Km. 36, Banjarbaru, Kalimantan Selatan, Indonesia
[1]1711016210020@mhs.ulm.ac.id
[2]muliadi@ulm.ac.id
[3]radityo.adi@ulm.ac.id
[4]dwikartini@ulm.ac.id
[5]rudy.herteno@ulm.ac.id

Abstract

Phishing is a form of cyber crime that harms other people and includes acts that are against the law. There are several approaches to combating phishing crimes, one of which is by classifying phishing websites using machine learning methods. The dataset used is a phishing websites dataset from the UCI Repository with 11055 data and 30 categorical features. The classifier method used is XGBoost. XGBoost is good for classifying data with categorical features, but the performance of this algorithm can still be improved. To overcome these problems, researchers used a hyper-parameter tuning solution. XGBoost has several hyper-parameters that can be configured to improve the performance of the model. The problem of identifying good values for hyper-parameters is called hyper-parameter tuning. The hyper-parameter tuning method used is Random Search which is then validated using 5-Fold Cross Validation for 30 iterations. The configured XGBoost hyper-parameters include n_estimators, max_depth, subsample and learning_rate. Testing on XGBoost without hyperparameter tuning obtained an accuracy of 95.34%. Testing on XGBoost with hyperparameter tuning obtained an accuracy of 97.69%. Hyper-parameter tuning with Random Search on XGBoost for phishing websites classification provides improved model performance at an accuracy of about 2.35%.

Keywords: *Phishing, XGBoost, Hyper-parameter Tuning, Random Search, Categorical Feature.*

1. Pendahuluan

Phising adalah salah satu bentuk *cyber crime yang* menggunakan website palsu dimana tampilannya mirip dengan website aslinya. Website palsu ini digunakan untuk memancing korban memasukkan informasi pribadi rahasia sehingga *phiser* dapat mencuri informasi rahasia tersebut. *Phising* merupakan tindak pidana yang sangat merugikan dan termasuk perbuatan yang melawan hukum. Kurangnya pengetahuan pengguna terhadap ciri-ciri website *phising* sehingga terjebak pada website palsu merupakan faktor penyebab terjadinya *phising* [1]. Terdapat beberapa pendekatan untuk memberantas kejahatan *phising*, salah satunya dengan melakukan klasifikasi terhadap situs *phising* menggunakan metode *machine learning*. Dataset yang umum digunakan yaitu dataset situs *phising* dari *UCI Repository* dengan 11055 data dan 30 fitur kategorial [2].

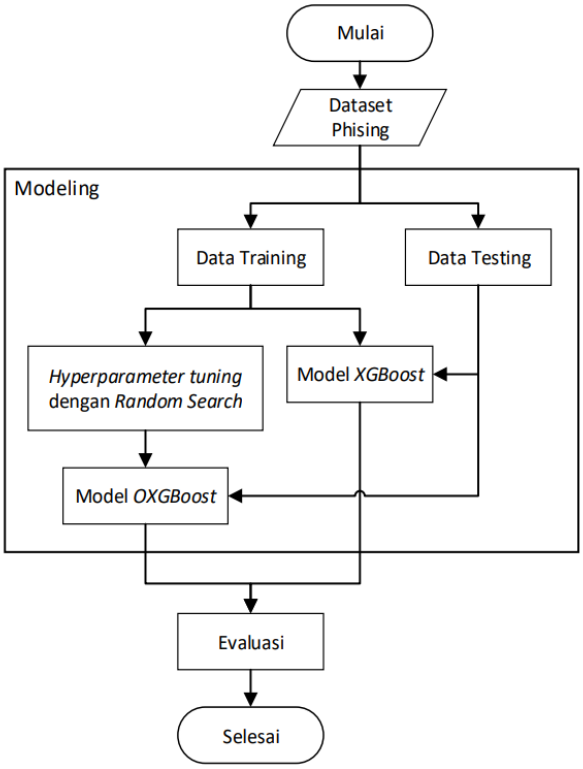
XGBoost merupakan algoritma klasifikasi berbasis *decision tree* yang menerapkan teknik *ensemble boosting. XGBoost* memiliki kinerja yang baik pada data dengan fitur kategorikal dan

tidak terlalu berpengaruh terhadap data dengan kelas tidak seimbang [3]. *XGBoost* memiliki keunggulan dalam hal kecepatan dan penggunaan memori. Pemanfaatan cache prosesor yang lebih baik, pemrosesan multicore dan komputasi paralel terdistribusi membuat sistem dapat berjalan lebih cepat daripada algoritma populer yang umum digunakan [4]. Kinerja dari *XGBoost* masih dapat ditingkatkan salah satunya dengan cara menyetel nilai *Hyper-parameter*. *XGBoost* merupakan salah satu algoritma yang memiliki banyak *hyper-parameter* dan konfigurasi *hyper-parameter* yang tepat dapat meningkatkan kinerja dari *XGBoost.* Masalah mengidentifikasi nilai yang baik untuk *hyper-parameter* disebut optimasi *hyper-parameter* atau *hyper-parameter tuning.*

Random search merupakan teknik *hyper-parameter tuning* yang memilih konfigurasi berdasarkan ruang *hyper-parameter* secara acak. Teknik ini sepenuhnya acak dan tidak menggunakan kecerdasan dalam memilih titik percobaan [5]. Kinerja model menggunakan *Random search* setara dengan kinerja yang diperoleh menggunakan teknik *hyper-parameter tuning meta heuristik* seperti *Genetic Algorithm, PSO* dan *EDA*, tetapi dengan komputasi yang lebih cepat dan efisien pada ruang *hyper-parameter* berdimensi tinggi, sehingga *random search* cocok diterapkan pada *XGBoost* yang memiliki banyak *hyper-parameter* [6].

2. Metodologi Penelitian

Alur dari penelitian dari klasifikasi situs *phising* menggunakan *XGBoost* dengan *hyper-parameter tuning* secara sistematis terdiri dari pengumpulan data, modeling dan evaluasi yang di presentasikan pada gambar 1.



Gambar 1. Desain alur penelitian.

2.1. Pengumpulan Data

Dataset pada penelitian ini menggunakan *Phising* Website Data Set dari *UCI Machine Learning Repository.* Dataset ini memiliki 11.055 instance dengan 30 fitur kategorikal dengan nilai -1, 0 dan 1 dan memiliki dua kelas yang terdiri dari 6.157 kelas *non-phising* yang dilambangkan dengan angka 1 dan 4.898 kelas *phising* yang dilambangkan dengan angka -1. Perbandingan persentase

kelas pada dataset ini yaitu kelas *non-phising* sebesar 55,7% dan kelas *phising* sebesar 44,3%, sehingga termasuk data yang seimbang. Dataset ini sudah bersih sehingga tidak dilakukan proses *preprocessing*.

2.2. Modeling

Tahapan awal dari modeling yaitu membagi data. 80% dari total data digunakan sebagai data training, dan 20% dari total data digunakan sebagai data testing. Pembagian data dilakukan secara stratifikasi, sehingga proporsi perbandingan kelas pada *data training* dan *data testing* sama. *Data training* berfungsi sebagai data untuk melatih model dan *data testing* merupakan representasi dari data masa depan yang akan diprediksi kelasnya. *Data testing* merupakan *unseen data* yang digunakan untuk mengetahui kinerja dari model yang telah dilatih.

Tabel 1. Pembagian Data

Data	Kelas *Phising*	Kelas Non-*Phising*	Total Instance
Data training	4.926	3.918	8.844
Data testing	1.231	980	2.211

Modeling pada penelitian ini terbagi menjadi dua percobaan yaitu pertama melakukan klasifikasi menggunakan *XGBoost* tanpa *hyper-parameter tuning*, dan yang ke dua melakukan klasifikasi menggunakan *XGBoost* dengan *hyper-parameter tuning* menggunakan *Random search*.

a. Klasifikasi *XGBoost* tanpa *Hyper-parameter Tuning*
Setelah dataset dibagi, selanjutnya sebanyak 8.844 *data training* digunakan untuk melakukan klasifikasi dengan *XGBoost* tanpa *hyper-parameter tuning*. Pada tabel 2 diuraikan penjelasan *hyper-parameter* yang terdapat pada *XGBoost*.

Tabel 2. *Hyper-parameter Default XGBoost*

Nama	Deskripsi	Nilai Default
n_estimator	Jumlah pohon individu (*tree boosting)*	100
max_depth	Kedalaman maksimum dari pohon individu	3
subsample	Rasio *instance* dari *data training* yang digunakan untuk membuat pohon individu	1
learning_rate	Penyusutan langkah yang digunakan dalam pembaruan model	0.1
gamma	*Loss reduction* minimum yang diperlukan untuk membuat partisi pada simpul daun dari pohon individu.	0
colsample_bytree	Rasio fitur dari *data training* yang digunakan untuk membuat pohon individu	1
min_child_weight	Bobot minimum yang diperlukan untuk membuat sebuah daun pada pohon individu	1

Klasifikasi dilakukan dengan melatih model *XGBoost* menggunakan seluruh *data training*, yaitu sebanyak 8.844 *instance*. Model *XGBoost* tanpa *hyper-parameter tuning* kemudian dievaluasi untuk mengetahui kinerja dari model.

b. Klasifikasi *XGBoost* dengan *Hyper-parameter Tuning*
Setelah dataset dibagi, selanjutnya selanjutnya sebanyak 8.844 *data training* untuk melakukan *hyper-parameter tuning* dengan *random search*. *Random search* merupakan teknik *hyper-parameter tuning* yang memilih konfigurasi berdasarkan ruang *hyper-parameter* secara acak. Teknik ini sepenuhnya acak dan tidak menggunakan kecerdasan

dalam memilih titik percobaan. Pada penelitian ini *Random search* dilakukan sebanyak 30 iterasi sehingga akan menghasilkan 30 kandidat *hyper-parameter*, kemudian setiap kandidat *hyper-parameter* akan divalidasi dengan 5-*Fold Cross Validation*. Konfigurasi *hyper-parameter* optimal dipilih berdasarkan nilai akurasi *cross validation* tertinggi dari semua kandidat yang telah dihasilkan. Pada tabel 3 dijelaskan *hyper-parameter XGBoost* yang dikonfigurasi menggunakan *random search*.

Tabel 3. Domain Pencarian *Hyper-parameter*

Nama	Deskripsi	Domain	
		Min	Maks
n_estimator	Jumlah pohon individu (*tree boosting)*	1	200
max_depth	Kedalaman maksimum dari pohon individu	1	10
subsample	Rasio *instance* dari *data training* yang digunakan untuk membuat pohon individu	0.25	0.75
learning_rate	Penyusutan langkah yang digunakan dalam pembaruan model	0.01	0.5

Konfigurasi *hyper-parameter* optimal kemudian digunakan untuk melatih ulang model *XGBoost* menggunakan keseluruhan dari *data training*. Model *XGBoost* dengan *hyper-parameter tuning* kemudian dievaluasi untuk mengetahui kinerja dari model.

2.3. Evaluasi

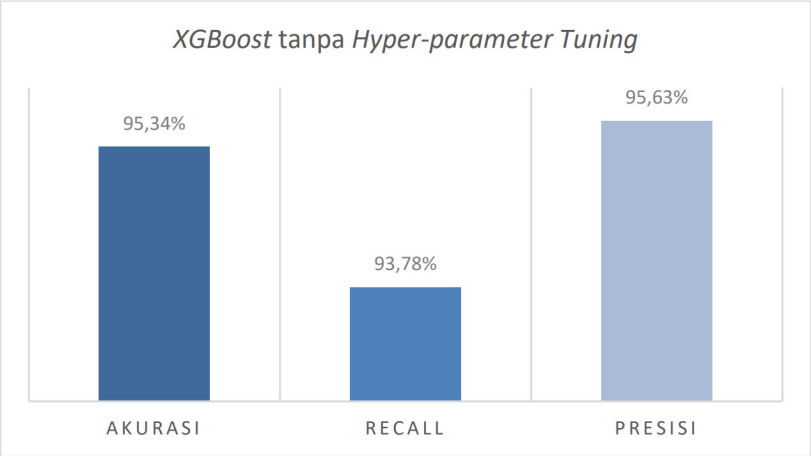
Evaluasi merupakan tahapan untuk menilai kinerja dari model yang dihasilkan. Untuk mengetahui kinerja dari model, pada penelitian ini model dievaluasi menggunakan akurasi, recall dan presisi.

3. Hasil Dan Pembahasan

3.1. Hasil

a. Klasifikasi *XGBoost* tanpa *Hyper-parameter Tuning*

Model *XGBoost tanpa hyper-parameter tuning* dilakukan training menggunakan *data training* kemudian model dievaluasi untuk mengetahui kinerja dari model. Adapun untuk hasil kinerja yang dihasilkan pada model *XGBoost* tanpa *hyper-parameter tuning* disajikan pada gambar 2.



Gambar 2. Kinerja model *XGBoost* tanpa *hyper-parameter tuning*

b. Klasifikasi XGBoost dengan Hyper-parameter Tuning

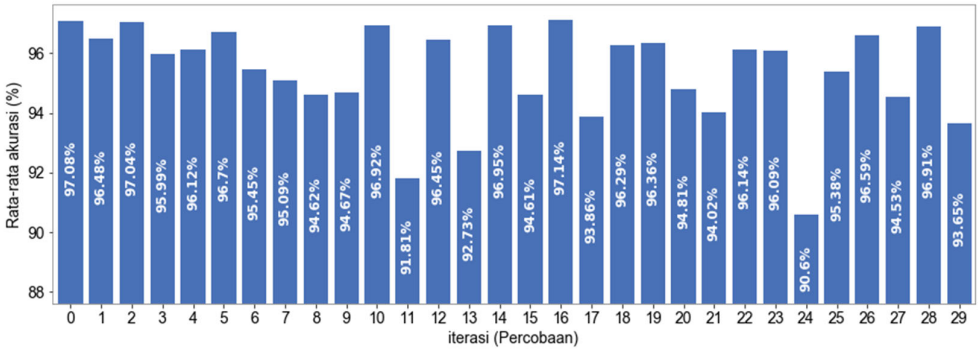
Hyper-parameter tuning dengan random search dilakukan sebanyak 30 iterasi sehingga menghasilkan 30 kandidat hyper-parameter secara acak. Pada tabel 4 disajikan kandidat hyper-parameter yang dihasilkan dari random search dengan 30 iterasi

Tabel 4. Kandidat *hyper-parameter*

Iterasi	Learning_rate	Max_depth	N_estimators	Subsample
0	0.193524658	8	189	0.548425079
1	0.228458049	7	75	0.479624446
2	0.173517219	8	152	0.575444236
3	0.037641674	8	158	0.356169555
4	0.099094234	5	161	0.402121121
5	0.267130651	9	49	0.51238733
6	0.205931876	3	108	0.507117219
7	0.300283139	3	51	0.590153769
8	0.230744633	2	132	0.721100878
9	0.286011227	10	9	0.257983126
10	0.123137975	7	172	0.497588455
11	0.026850375	1	164	0.341118044
12	0.380127091	6	54	0.52335514
13	0.100578683	2	44	0.719749471
14	0.448465402	8	190	0.412665165
15	0.200451872	2	111	0.664368755
16	0.18480913	9	157	0.65109849
17	0.046529815	7	9	0.636122385
18	0.107370684	8	63	0.657730714
19	0.356360098	3	163	0.635635173
20	0.046281879	7	41	0.707479838
21	0.426518903	2	33	0.281779175
22	0.162381338	8	37	0.614803089
23	0.322403161	3	193	0.441463437
24	0.486138927	3	1	0.610864761
25	0.125632611	7	27	0.511366415
26	0.219495099	10	124	0.303945713
27	0.025400301	7	52	0.531637786
28	0.350802882	5	151	0.55220869
29	0.274522135	4	13	0.288489955

Untuk menentukan konfigurasi *hyper-parameter* mana yang paling optimal, setiap konfigurasi dilakukan validasi secara 5-Fold Cros Validation menggunakan seluruh *data training*. Adapun hasil dari 5-Fold Cross Validation dari setiap kandidat disajikan dalam bentuk diagram batang pada gamberGambar **3**.

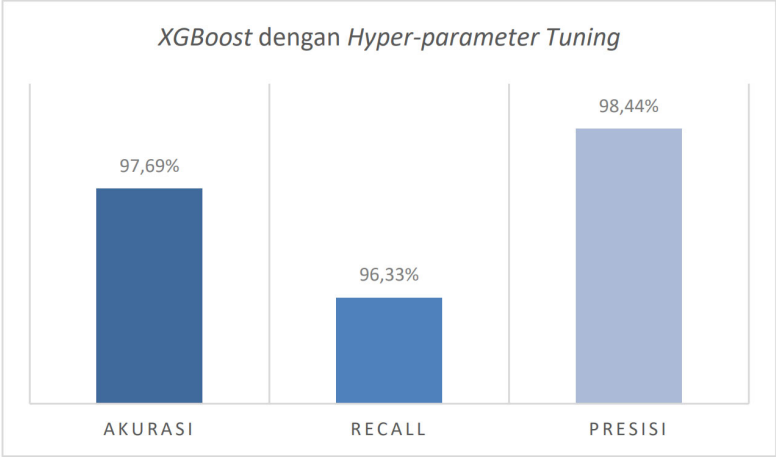
p-ISSN: 1979-5661
e-ISSN: 2622-321X



Gambar 3. Hasil dari *Random search* dengan 30 iterasi

Pada grafik diatas, garis vertikal mewakili rata-rata akurasi dari kandidat *hyper-parameter* atau validation score, sedangkan garis horizontal mewakili iterasi atau percobaan ke *n* dari *random search*. Dari hasil *hyper-parameter tuning* menggunakan *random search* didapat akurasi 5-*Fold Cross Validation* tertinggi pada iterasi ke 16 yaitu sebesar 97.14%, dengan nilai learning rate=0.18480913, max depth=9, n_estimator=157, subsample=0.65109849.

Setelah didapat nilai *hyper-parameter* optimal menggunakan *random search.* Model *XGBoost dengan hyper-parameter tuning* kemudian dilakukan final training menggunakan *data training* beserta dengan konfigurasi *hyper-parameter* optimal, kemudian model dievaluasi untuk mengetahui kinerja dari model. Adapun untuk hasil kinerja yang dihasilkan pada model *XGBoost* dengan *hyper-parameter tuning* disajikan pada gambar 4.



Gambar 4. Kinerja model *XGBoost* dengan *hyper-parameter tuning*

3.2. Pembahasan

Pada percobaan klasifikasi *XGBoost* tanpa *hyper-parameter tuning*, model *XGBoost* tanpa *hyper-parameter tuning* dilatih menggunakan 8.844 *data training* dan dievaluasi menggunakan 2.211 *data testing*. Hasil kinerja dari klasifikasi *XGBoost* tanpa *hyper-parameter tuning* yaitu diperoleh akurasi sebesar 95,34%, recall sebesar 93,78% dan presisi sebesar 95,63%.

Pada percobaan klasifikasi *XGBoost* dengan *hyper-parameter tuning*, 8.844 *data training* digunakan untuk melakukan *hyper-parameter tuning* dengan *random search*. *Random search* dilakukan sebanyak 30 iterasi sehingga akan menghasilkan 30 kandidat *hyper-parameter*, kemudian setiap kandidat *hyper-parameter* divalidasi secara 5-Fold Cross Validation. Nilai *hyper-*

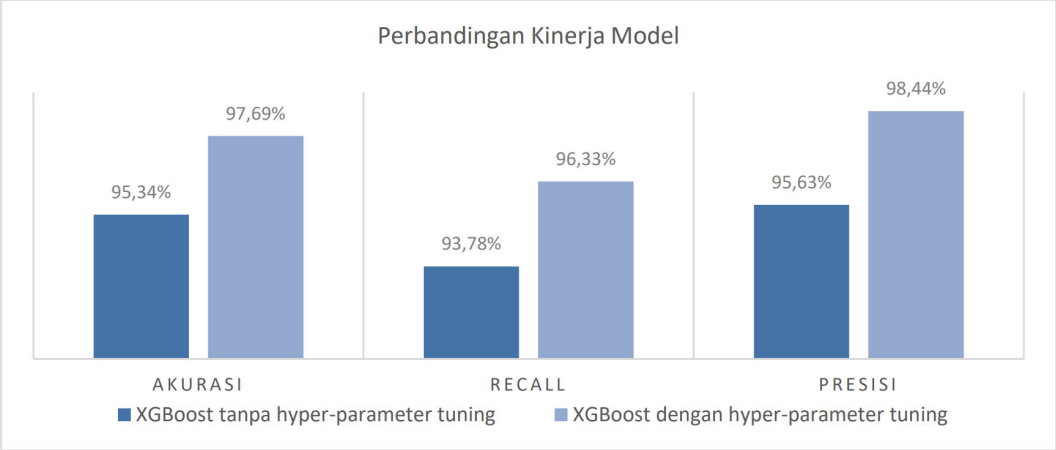
parameter optimal dipilih berdasarkan kandidat dengan nilai validasi tertinggi. *Random search* dilakukan sebanyak 30 iterasi karena tidak mendapatkan hasil yang lebih baik ketika digunakan iterasi yang lebih dari 30. Pada tabel 5 disajikan perbedaan konfigurasi *hyper-parameter* tanpa *hyper-parameter tuning* dan *hyper-parameter tuning.*

Tabel 5. Perbedaan Konfigurasi *Hyper-parameter*

Model	Learning_rate	Max_depth	N_estimators	Subsample
Tanpa *hyper-parameter tuning*	0.1	3	100	1
Dengan *hyper-parameter tuning*	0.18480913	9	157	0.65109849

Model *XGBoost* dengan *hyper-parameter tuning* kemudian dilakukan final training, yaitu model *XGBoost* dengan *hyper-parameter tuning* dilatih menggunakan 8.844 *data training* dan dievaluasi menggunakan 2.211 *data testing*. Hasil kinerja dari klasifikasi *XGBoost* dengan *hyper-parameter tuning* yaitu diperoleh akurasi sebesar 97,69%, recall sebesar 96,33% dan presisi sebesar 98,44%.

Untuk mengetahui pergaruh *hyper-parameter tuning* pada kinerja *XGBoost* dalam melakukan klasifikasi pada dataset situs *phising*, maka dilakukan perbandingan antara kinerja model *XGBoost* dengan *hyper-parameter tuning* dengan kinerja model *XGBoost* dengan *hyper-parameter tuning.* Pada Gambar 5 dapat dilihat grafik perbandingan anatara kinerja model *XGBoost* tanpa *hyper-parameter tuning* dengan kinerja model *XGBoost* dengan *hyper-parameter tuning.*



Gambar 5. Perbandingan kinerja model

4. Kesimpulan

Klasifikasi situs *phising* menggunakan *XGBoost* tanpa *hyper-parameter tuning* mendapatkan akurasi sebesar 95,34%, recall sebesar 93,78% dan presisi sebesar 95,63%. Klasifikasi situs *phising* menggunakan *XGBoost* tanpa *hyper-parameter tuning* mendapatkan akurasi sebesar 97,69%, recall sebesar 96,33% dan presisi sebesar 98,44%. *Hyper-parameter tuning* dengan *random search* pada *XGBoost* untuk klasifikasi situs *phising* memberikan peningkatan kinerja pada akurasi sekitar 2,35%, pada recall mengalami peningkatan sekitar 2,55% dan pada presisi mengalami peningkatan sekitar 2,81%.

Daftar Pustaka

[1]. A. S. Gulo and K. Nawawi, "Cyber Crime dalam Bentuk *Phising* Berdasarkan Undang-Undang Informasi dan Transaksi Elektronik" *PAMPAS: Journal of Criminal,* vol. 1, no. 2, p. 68-81, 2020.

[2]. R. M. Mohammad, F. Thabtah and L. McCluskey, "Intelligent rule-based phishing websites classification*"* IET Information Security, vol. 8, p.153-160, 2013

[3]. R. Pavan, M. Nara, S. Gopianth and N. Patil, "Bayesian Optimization and Gradient Boosting to Detect Phishing Websites" in *2021 55th Annual Conference on Information Sciences and Systems (CISS)*, Baltimore, MD, USA.

[4]. T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System" in *KDD '16: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* San Francisco California, USA, 2016, pp. 785-794.

[5]. J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization" *Journal of Machine Learning Research,* vol. 13, no. 10, p. 281-305, 2012.

[6]. R. G. Mantovani, A. L. D. Rossi, J. Vanschoren, B. Bischl and A. C. P. L. F. Carvalho, "Effectiveness of Random Search in SVM Hyper-parameter Tuning" in *2015 International Joint Conference on Neural Networks (IJCNN),* Killarney, Ireland, 2015.