UNITECH SELECTED PAPERS

INTERNATIONAL SCIENTIFIC CONFERENCE
20-22 November 2025, GABROVO

ACCELERATING SVM HYPERPARAMETER TUNING FOR PHISHING WEBSITE DETECTION USING HIGH-PERFORMANCE COMPUTING

Biljana Lakić[1*], Kristijan Kuk[2]

[1]University of Criminal Investigation and Police Studies, Cara Dušana 196 Zemun, Belgrade, Serbia
[2]University of Criminal Investigation and Police Studies, Cara Dušana 196 Zemun, Belgrade,
**Corresponding author: biljana.lakic@kpu.edu.rs*

Abstract
Phishing attacks continue to pose a significant threat to digital security, necessitating the development of more effective detection mechanisms. This study explores the optimization of widely used machine learning classifier, Support Vector Machines (SVM), for the task of phishing website detection, leveraging the computational capabilities of Serbia's National AI Platform. The research focuses on hyperparameter tuning using optimization techniques executed in high-performance DGX-A100 nodes. By utilizing the FastML Engine within the Codex AI SUITE, the study achieves scalable orchestration of large-scale experiments, enabling rapid evaluation of numerous hyperparameter configurations. Results demonstrate substantial improvements in training time, highlighting the potential of HPC resources to enhance cybersecurity applications and support the strategic utilization of national AI infrastructure.
Keywords: machine learning, classification, high-performance computing, optimization.

INTRODUCTION

In today's digital environment, phishing attacks represent one of the most widespread and dangerous forms of cyber threats. The goal of these attacks is to deceive users into revealing confidential information, most often through fake websites that mimic legitimate services. Given the increasing sophistication of such attacks, traditional detection methods are becoming insufficient, which opens the door for the application of advanced machine learning techniques. The most common anti-phishing technique, blacklisting, is not enough to protect users from phishing because new phishing sites appear every day in thousands.

Machine learning classifiers, such as Support Vector Machines (SVM), have proven to be effective tools in the automatic detection of phishing websites. However, their performance largely depends on the proper selection of hyperparameters, which directly influence the model's effectiveness. The process of hyperparameter optimization can be extremely demanding, especially when dealing with complex models and large datasets.

In this context, leveraging the resources of Serbia's National AI Platform, including a supercomputer with high-capacity parallel processing capabilities, offers a unique opportunity to accelerate and enhance the optimization process. This infrastructure enables the evaluation of a large number of hyperparameter combinations within a significantly shorter time frame, thereby creating space for application of advanced algorithms such Bayesian optimization and genetic algorithms.

The aim of this study is to investigate the impact of various hyperparameter optimization methods on the performance of

doi: www.doi.org/10.70456/.....

classifier in the task of phishing website detection. Special emphasis will be placed on the efficiency, scalability, and accuracy of the models in the context of using HPC resources. The results of this research may contribute to the development of more robust and faster cybersecurity systems, as well as to the improved utilization of domestic AI infrastructure.

To promote the development and application of artificial intelligence technologies in public web services as AI-based phishing websites or similarity cybersecurity AI-driven products, across the entire industrial sector of the Republic of Serbia, and to provide strategic support to startup companies in building innovative AI-driven products, the Government of the Republic of Serbia has established the National Platform for Artificial Intelligence (AI Platform) [1].

The establishment and development of this platform is guided by the goals and measures outlined in the Serbia's Artificial Intelligence Development Strategy for the periods 2020-2025 and 2025-2030.

The AI Platform is deployed as a high-performance supercomputing infrastructure located within The State Data Center in Kragujevac, which adheres to the highest reliability standards.

ARCHITECTURE OF THE NATIONAL AI PLATFORM

The system of the National AI Platform is built by integrating the computational power of four DGX-A100 servers, equipped with a total of 32 NVIDIA A100 GPUs, optimized for deep learning (DL) and multi-node HPC simulations. The DGX-A100 servers are fully dedicated to AI workloads, as the solution includes separate nodes for system administration and user access [2].

All servers are interconnected via two Mellanox InfiniBand HDR networks, providing high bandwidth and access to a centralized, optimized AI data management system. NVIDIA DGX A100 is a universal platform for the entire AI infrastructure, from analytics and training to inference. It sets a new standard in computational density, delivering 5 petaFLOPS of AI performance in 6U unit, replacing fragmented infrastructure silos with a unified platform for any AI computing task.

NVIDIA DGX A100 is the world's first AI system built on NVIDIA A100 Tensor Core GPU architecture. By integrating eight A100 GPUs, the system delivers unmatched acceleration and is fully optimized for NVIDIA CUDA-X^{tm} software and the NVIDIA data center stack [2].

Key innovations include [2]:

- TF32 precision, which operates like FP32 but delivers up to 20x more AI FLOPS compared to the previous generation, without requiring code changes.
- Automatic mixed precision with FP16, offering an additional 2x performance boost with just one extra line of code.
- Memory bandwidth of 1.6 TB/s, a 70% increase over the previous generation.
- On-chip memory enhancements, including 40 MB of L2 cache, nearly 7x larger, maximized computational throughput.

This unprecedented power enables the fastest time-to-solution for training, inference, and analytics, empowering users to tackle challenges that were previously impractical or impossible to solve.

Codex AI Suite - FastML

To support large-scale machine learning workflows, this research utilizes the Codex AI SUITE – FastML Engine (Atos) [3], an advanced Data Science and Artificial Intelligence platform designed for high-performance AI computing environments. Its primary goal is to simplify access to machine learning environments, streamline model development, and enable deployment across clusters of AI compute nodes, particularly within infrastructures such as DGX POD and SUPERPOD.

FastML Engine abstracts the technical complexity of supercomputing, allowing

doi: www.doi.org/10.70456/.....

ML developers to manage the entire production workflow via command-line interface or web browser.

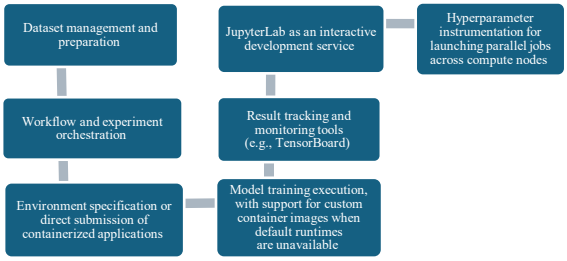

Fig.1. *Block diagram of the machine learning workflow in HPC environment*

The platform is fully integrated with the NVIDIA DGX software stack and supports NVIDIA GPU Cloud (NGC) images and artifacts, including optimized ML frameworks and libraries packaged in Docker containers.

FastML Engine also utilizes a hybrid orchestration layer capable of interacting with job schedulers and deploying interactive sessions on dedicated service nodes, enabling parallel execution of workloads alongside code development.

The graphical interface further enhances usability by allowing researchers to launch JupyterLab notebook environments preloaded with selected ML frameworks. Users can mount datasets directly into their notebook instances and access their home directories, with deployment options available for both service and compute nodes. A dedicated GUI page facilitates comprehensive management of JupyterLab sessions.

This infrastructure provides the computational foundation for executing large-scale hyperparameter optimization experiments and model training tasks, contributing to the overall efficiency and reproducibility of the research.

MODEL DEVELOPMENT CLASSIFFICATION ALGORITHMS FOR PHISHING DETECTION

For model development, the PhiUSIIL Phishing URL dataset [4] was used, obtained from the UC Irvine Machine Learning Repository. This dataset was created by Arvind Prasad and Shalini Chandra from Babashaheb Bhimrao Ambedkar University in 2024, with the aim of supporting the creation of efficient and effective frameworks for phishing detection. The original corpus includes 134,850 legitimate and 100,945 phishing URLs (a total of 235,795 instances), with no missing values, extracted from the source code of web pages and URLs. Label 1 corresponds to legitimate URL, while label 0 indicates a phishing URL.

The dataset contains a total of 54 features (variables), of which 3 are textual, 31 are numerical, and the remaining 20 are nominal, dichotomous variables that take values of 0 or 1. For the construction of our model, only the features with numerical values were used.

Classification algorithms

Support Vector Machine (SVM) is one of the most widely used supervised learning algorithms, applicable to both Classification and Regression tasks. Its primary use lies in solving classification problems within the field of machine learning. The main objective of the SVM algorithm is to identify an optimal decision boundary that separates an n-dimensional feature space into distinct classes, allowing new data points to be accurately categorized [5]. This optimal boundary is known as a hyperplane. To construct the hyperplane, SVM selects critical data points, those that exert the greatest influence on the boundary's position. These points are referred to as support vectors, which is where the algorithm gets its name [6]. In Support Vector Machine algorithm, kernel functions play a crucial role in transforming data into higher-dimensional space in order to find the optimal hyperplane for separating classes. The main types of kernel function [7] commonly used are:

- Linear kernel - Lk,
- Polynomial kernel - Pk,
- Radial Basis Function (RBF) or

doi: www.doi.org/10.70456/.....

Gaussian kernel - Gk,
- Sigmoid kernel - Sk.

EXPERIMENTAL ANALYSIS OF CLASSIFICATION ALGORITHMS

The significant impact of hyper-parameters on model performance in most machine learning algorithms requires careful tuning [10]. Instead of relying on a manual trial-and-error approach, which is often time-consuming and difficult to reproduce, automated hyperparameter optimization methods can be employed [11]. These methods, based on error estimates through resampling in supervised learning, enable efficient identification of optimal configurations [12].

As part of this research, a series of programs were developed using the Python programing language [13], leveraging its rich and diverse set of libraries that support efficient implementation of machine learning algorithms and precise performance evaluation. Python was selected for its flexibility, readable syntax, and extensive support for scientific and engineering applications, including libraries such as scikit-learn, NumPy, pandas, and time.

After implementation, the code was executed within the Visual Studio development environment, which enabled detailed monitoring of the model training process and measurement of execution time for each experiment. Execution time was recorded consistently, providing an additional dimension of evaluation regarding algorithm efficiency and the impact of different hyperparameter configurations on processing speed.

In addition to local execution, the same code was tested within the FastML-e environment, which supports distributed and randomized processing. Special attention was given to tracking execution time under parallel processing conditions, allowing for a comparative analysis between manual and automated approaches to hyperparameter optimization.

The experiments encompassed all relevant hyperparameters of the SVM algorithm, focusing on various kernel types: linear, polynomial, Gaussian and sigmoid. For each kernel, evaluation was conducted using two approaches: the first involved, while the second employed parallel testing of multiple configurations using automated optimization techniques. This dual approach enabled a comprehensive analysis of the influence of individual hyperparameters on model performance and facilitated the identification of optimal combinations for further experimental and applied purposes.

The results obtained throughout research are presented in tabular form to ensure clarity and facilitate analysis.

It is well known that, regardless of the type of model being used, the dataset must be divided into two groups: one for the training, and the other for the evaluation of the model performance. In this case, we divided 30% of our data as test set and 70% as train set.

Table 1. presents the results obtained by applying the SVM algorithm with different kernel functions. Each kernel was used to train model separately, and the evaluation encompassed metrics such as accuracy, precision, recall, F1-score, and execution time.

	SVM kernel	*Precision*	*Recall*	*F1-score*	*Support*
0	Lk, Gk, Pk, Sk	1.00 1.00 1.00 0.99	1.00	1.00 1.00 1.00 0.99	30151
1		1.00	1.00	1.00	40588
Accuracy	Lk, Gk, Pk, Sk			1.00	70739
Macro avg	Lk, Gk, Pk, Sk	1.00	1.00	1.00	70739
Weighted avg	Lk, Gk, Pk, Sk	1.00	1.00	1.00	70399

Table 1. *Comparative performance metrics of the SVM algorithm kernels*

Table 2. presents results of four confusion matrices and illustrates the classification performance of each kernel. These matrices reveal that the poorest results are produced by the sigmoid kernel. Each matrix displays the number of correctly and incorrectly classified instances, representing the relationships between true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) predictions [14]. These results clearly show that the sigmoid kernel yields the least accurate classification outcomes in this experiment. In contrast, the linear and Gaussian kernel exhibit the

doi: www.doi.org/10.70456/.....

highest precision and stability, making them more suitable for binary classification tasks in this context.

SVM kernel	Lk	Gk	Pk	Lk	Gk	Pk
	Positive			Negative		
Positive	30136	30147	30119	4	15	32
Negative	0	0	0	40588	40588	40588

***Table 2**. Confusion matrices*

Figure 2. presents execution times for the SVM algorithm using different kernel functions across two computing platforms: a regular PC and a high-performance computing system (HPC). The values expressed in seconds represent the duration of model training for each kernel. It is evident that HPC platform significantly reduces execution time compared to the regular PC, particularly the advantages of parallel processing and optimized resources in high-performance environments. The regular PC used in the experiment is equipped with a single processor (Intel Celeron) and 12 GB of RAM. In contrast, the experimental container within the HPC environment is configured with four CPUs and 8 GB of RAM, enabling parallel processing and faster algorithm execution.

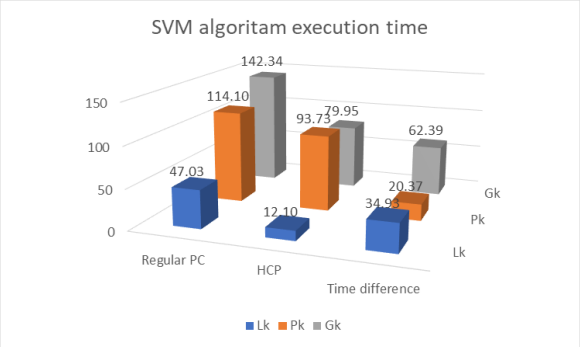

***Fig. 2.** Execution time of SVM algorithm based on hyperparameter tuning method*

To determine the optimal parameters for the SVM algorithm, we applied the build-in Grid Search - GridSearchCV and Randomized Search (Random Search) - RandomSearchCV functions available in the scikit-learn Python library [15], specifically within the sklearn.model_selection module. The primary purpose of Grid Search function is to enable parallel testing of parameters across a wide range of values, which facilitates the identification of the best configuration for model construction. In other words, Grid Search is a specialized hyperparameter optimization technique that exhaustively evaluates all possible parameter combinations to find the most suitable one [16]. Random Search also tests combinations of hyperparameters from hyperparameter space to determine the optimal subset, where instead of trying all possible combinations, Random Search randomly selects a specific number of combinations from a distribution of hyperparameter values.

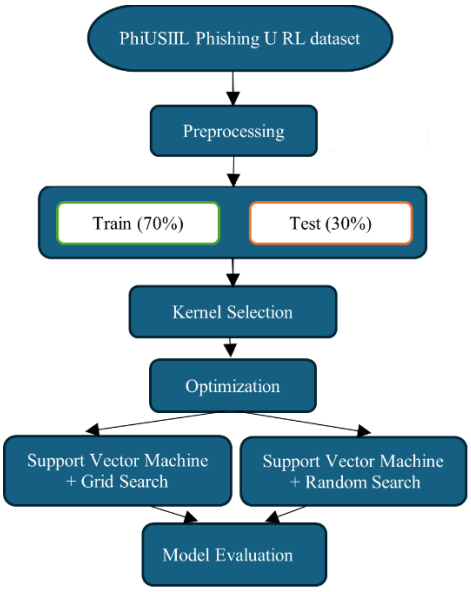

***Fig 3**.Process Flow of the Proposed Method*

Cross-validation is a very important tool for model validation in machine learning [17]. Cross-validation is used both for tuning and evaluating hyperparameters, as well as for comparing a set of models to select the most suitable one [18]. In this case, we used value of 5 for Cross-validation, which means that model will iterate learning 5 times, and that dataset is divided into 5 parts and 5 folds. That means that in the first iteration we used data from fold 1, in the second iteration data from fold 2, and so on. However, the technique requires more execution time, which makes the ability to

doi: www.doi.org/10.70456/.....

use HPC resources extremely important for saving time.

We tested Lk, Gk, and Pk, also different values for C parameter, and execution time for each procedure. The results obtained are presented in Table 3. The linear kernel emerged as the most optimal choice for model construction, demonstrating superior performance both in the application of the Grid Search algorithm and the Random Search procedure. Random Search showed better execution time.

SVM	C	Kernel	Time (s)
SVM-grid	10	Lk	7114.95
SVM-random	10	Lk	4560.08

***Table 3**. Best C parameter value, kernel and execution time*

Based on the above results, this approach (Figure 3.) is a viable solution to help overcome the shortcomings of the SVM algorithm hyperparameter testing.

CONCLUSION

Results of experiment shown in these papers clearly highlight significant differences in execution time between the standard computing platform and the high-performance computing (HPC) environment. Moreover, the results indicate that the optimal solution for building the best model to detect phishing websites is the application of the SVM algorithm with a linear kernel, which achieved the best performance. These findings suggest that future research should focus on parallel parameter testing, which would enable the selection of the most effective model for detecting phishing websites with significant time savings using HPC resources. Such an approach would further facilitate the deployment of the model for real-time threat detection.

Acknowledgments: Authors gratefully acknowledges the support of the Serbian National AI Platform, whose resources and infrastructure contributed to the successful completion of the work. Also, we express deep gratitude to Arvind Prasad and Shalini Chandra from Babashaheb Bhimrao Ambedkar University for their contribution to our research through The PhiUSIIL Phishing URL dataset. Their work and the availability of the dataset were crucial for the development of our model and the training process in this paper.

REFERENCE

[1] Nacionalna platforma za veštačku inteligenciju, "Nacionalna VI platforma." Accessed: Oct. 19, 2025. [Online]. Available: https://www.ai.gov.rs/tekst/sr/189/nacionalna-vi-platforma.php

[2] NVIDIA, "Introduction to the NVIDIA DGX A100 System." Accessed: Oct. 19, 2025. [Online]. Available: https://docs.nvidia.com/dgx/dgxa100-user-guide/introduction-to-dgxa100.html

[3] M. Vizard, "Atos Launches AI Codex Suite to Accelerate AI App Development," *IT Business Edge*, Jun. 2018, Accessed: Oct. 19, 2025. [Online]. Available: https://www.itbusinessedge.com/business-intelligence/atos-launches-ai-codex-suite-to-accelerate-ai-app-development/

[4] A. Prasad and S. Chandra, "PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning," *Comput Secur*, vol. 136, p. 103545, Jan. 2024, doi: 10.1016/j.cose.2023.103545.

[5] K. S. Chong and N. Shah, "Comparison of Naive Bayes and SVM Classification in Grid-Search Hyperparameter Tuned and Non-Hyperparameter Tuned Healthcare Stock Market Sentiment Analysis," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 12, 2022, doi: 10.14569/IJACSA.2022.0131213.

[6] JavaTpoint, "Support Vector Machine Algorithm," Java Tpoint.

[7] N. Guenther and M. Schonlau, "Support vector machines," *Stata Journal*, vol. 16, no. 4, 2016, doi: 10.1177/1536867x1601600407.

[8] P. Probst, M. N. Wright, and A. L. Boulesteix,

doi: www.doi.org/10.70456/.....

"Hyperparameters and tuning strategies for random forest," 2019. doi: 10.1002/widm.1301.

[10] W. Pannakkong, K. Thiwa-Anont, K. Singthong, P. Parthanadee, and J. Buddhakulsomsiri, "Hyperparameter Tuning of Machine Learni ng Algorithms Using Response Surface Met hodology: A Case Study of ANN, SVM, and DBN," *Math Probl Eng*, vol. 2022, 2022, doi: 10.1155/2022/8513719.

[11] E. Elgeldawi, A. Sayed, A. R. Galal, and A. M. Zaki, "Hyperparameter tuning for machine learning algorithms used for arabic sentiment analysis," *Informatics*, vol. 8, no. 4, 2021, doi: 10.3390/informatics8040079.

[12] B. Bischl *et al.*, "Hyperparameter optimization: Foundations, algorithms, best practices, and open challen ges," 2023. doi: 10.1002/widm.1484.

[13] S. Raschka, J. Patterson, and C. Nolet, "Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence," 2020. doi: 10.3390/info11040193.

[14] S. Swaminathan and B. R. Tantri, "Confusion Matrix-Based Performance Evaluation Metrics," *Afr ican Journal of Biomedical Research*, vol. 27, pp. 4023–4031, Nov. 2024, doi: 10.53555/AJBR.v27i4S.4345.

[15] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, 2011.

[16] C. G. Siji George and B. Sumathi, "Grid search tuning of hyperparameters in ra ndom forest classifier for customer feedback sentiment prediction," *International Journal of Advanced Computer Science and Applicat ions*, vol. 11, no. 9, 2020, doi: 10.14569/IJACSA.2020.0110920.

[17]A. Seraj et al., "Cross-validation," in Handbook of HydroInformatics: Volume I: Classic Soft-Computing Techniques, 2022. doi: 10.1016/B978-0-12-821285-1.00021-X.

[18] L. A. Yates, Z. Aandahl, S. A. Richards, and B. W. Brook, "Cross validation for model selection: A review with examples from ecology," Ecol Monogr, vol. 93, no. 1, 2023, doi: 10.1002/ecm.1557.

doi: www.doi.org/10.70456/.....