Predicting Counter-Strike 2 Match Outcomes Using Machine Learning

Benedictus Bellion Leander
Computer Science Department
School of Computing and Creative Arts
Bina Nusantara University,
Jakarta, Indonesia 11480
benedictus.leander@binus.ac.id

Christian Antonio Yotanka Tampubolon
Computer Science Department
School of Computing and Creative Arts
Bina Nusantara University,
Jakarta, Indonesia 11480
christian.tampubolon001@binus.ac.id

Christopher Darren Muljono
Computer Science Department
School of Computing and Creative Arts
Bina Nusantara University,
Jakarta, Indonesia 11480
christopher.muljono001@binus.ac.id

Nunung Nurul Qomariyah
Computer Science Department
School of Computing and Creative Arts
Bina Nusantara University
Jakarta, Indonesia 11480
nunung.qomariyah@binus.edu

Raymond Bahana
Computer Science Department
School of Computing and Creative Arts
Bina Nusantara University
Jakarta, Indonesia 11480
rbahana@binus.edu

***Abstract*—In the world of multiplayer First-Person Shooter games, Counter-Strike stands out as one of the largest and long-lasting franchises in gaming. Correspondingly, its competitive scene continues to rise with the introduction of its latest iteration to date, Counter-Strike 2. This evolution highlights the need for optimal and efficient methods to analyze gameplay and improve player performance. Machine learning is utilized in this research to predict the outcomes of Premier Counter-Strike 2 matches through analyzing features related to the game, such as teams, players, historical match data, maps, win rate, and other in-game metrics collected. We utilized data derived from CSStats and tested models with classification algorithms such as support vector machines, logistic regression, random forest, and K-Nearest Neighbors for predictive accuracy. This study aims to build a system that helps players to analyze which areas and skills they need to improve, professional teams and coaches to optimize their strategy, as well as spectators to set better predictions on which teams will win or lose. Our findings show that Logistic Regression achieved the best performance, with an average accuracy of 92%. We confirmed these results through evaluation metrics such as recall, precision, F1-score, and ROC-AUC curve. Overall, this study seeks to enhance how teams and players optimize their play-making and strategy through machine learning in CS2.**

***Index Terms*—Counter-Strike 2, Match Outcome Prediction, Machine Learning, Classification Algorithms**

I. Introduction

Over the past few decades, Counter-Strike has grown from a small Half-Life mod made in the GoldSrc engine to a leading franchise in the world of First-Person Shooter games. The latest iteration, Counter-Strike 2, is the updated version of Counter-Strike: Global Offensive. Two teams—Terrorists (T) and Counter-Terrorists (CT)—compete in progressing rounds to either plant or defuse the bomb or to eliminate the opposing team. Players earn money at the end of each round based on their performance. Each player has a limited amount of money which can be used to buy weapons, armor, or grenades. This emphasizes the importance of strategy and tactics for the team to win the match. Counter-Strike 2's competitive scene is rooted in match preparation, strategy, as well as play-making and execution from the players. Hence, data analysis is crucial for improving strategic performance. Due to the complex factors that influences the outcome of the games, such as player performance, strategy planning, map preferences, and team coordination and dynamics, relying on intuition leave room for inaccuracies and biased predictions. The goal of this project is to create a machine learning model that predicts the outcomes of competitive matches in Counter-Strike 2. This project analyzes various features related to the teams, players, their historical match data, maps, win-rate, in-game metrics (e.g. Average Damage per Round - ADR), and more to predict whether a specific team will win or lose against the other. This system offers useful insight for players, teams, and spectators to make more informed decisions.

II. Related Work

A. Football match prediction

A data-driven model that accurately predicts football match outcomes by examining models such as Random Forest, Logistic Regression, and linear support vector classifiers. The features include team performance, player statistics, and historical match data. Models such as Random Forest combine multiple weaker models to create a stronger predictor and predict the outcome by following different paths based on tactical decision-making, recognizing strengths and weaknesses, choosing formations and players, and validating transfer targets. The study used data from the 2022-2023 English Premier League season. The binary class logistic regression model achieved the greatest success, generating a 7.57% return.

Accurate results can be achieved for players and betters to make informed decisions [1].

B. NBA (National Basketball Association) match prediction

This study proposes a new learning model for the prediction of NBA match outcomes using machine learning methods. Two indexes, team-efficiency index and individual-efficiency index are utilized for the machine learning process which determine the win function and decide the potential game outcome.

Factors such as player performance, home court advantage, and win/lose records observed from two NBA seasons (2013/2014-2017/2018) over a period of five years are taken into consideration, in order to prepare the proposed model. OTW (optimal time window), which is the time window within the training data set's time span, and whether the game was a practice or an official match, are influential in determining the accuracy of the model. This model uses supervised learning algorithms and has an average accuracy of 66%. The results from the model are utilized by players and coaches to strategically prepare for matchups and analyzing players' strengths and weaknesses [2].

C. League of Legends Match Prediction

This study aims to predict the outcome of the Riot Games title, *League of Legends*. Instead of using a Random Forest model, it uses the machine learning technique using a gradient-boosted tree as well as logistic regression. The categorical pre-game data is converted into a feature vector for machine learning models using one-hot encoding.

Features such as champions picked, player role information, the players' mastery level for their champions, as well as in-game player statistics are evaluated. The study uses data from the public Riot Games API via the Python Wrapper, Cassiopeia. Overall, the model used in this study has a 95% accuracy, which is in line with the industry standard [3].

III. Methodology

A. Data Collection

Most of the data required to train the models will be collected from CSStats, a website that tracks CS2 player's stats. This site features a table of recently finished player matches that updates every moment. Clicking on each item will navigate the user to a detailed page which provides in-depth information on that specific match. To extract the necessary data from the website, Selenium—an automation library used to extract and interact with information on a website—will be utilized to create a scraper. A drawback with CSStats is that the table only shows the last 50 player matches as the rest of the matches are stored within a database hidden to users. Unless permission is granted by the administrators to access this database, the scraper must refresh the site every few minutes to ensure all matches were captured and recorded.

Given the various types of player matches that exists in CS2, it is crucial to select one with consistent and relevant information for the model. In this case, 'Premier' competitive matches will primarily be scraped from the aforementioned table. These matches focus on high-level combat which provides a more refined and professional competitive experience. This is key to minimize abnormalities in data, as well as to warrant consistency. Additionally, this mode provides a robust elo-rating system which can be an important feature to train the model. The data collection process consisted of two phases—match data collection and player data collection.

1) **Match Data Collection**: Extracts the match-ids from each match, which will be used to access the necessary page that provides the detailed data and statistics of the specific match. Entries will be recorded and stored as a CSV file
2) **Player Data Collection**: Using the recorded match-ids from phase 1 to navigate the specified match page. This page provides a table which includes each player from their respective teams. Clicking on each player will navigate to the individual's stat page consisting of crucial information such as current rating, highest rating, k/d ratio, win-rate, and more. These are the data which will be scraped by the scraper. Entries will be recorded and stored as a CSV file.

B. Dataset Preprocessing

Preprocessing is necessary to minimize data abnormalities and outliers, which can negatively impact the model's performance [4]. Excessive noise in the data could cause the model to overfit or underfit. The scraped recorded data is relatively clean due to the amount of filters we add during the data collection phases. However, there are still a number of entries that are not proper for the model. For example, a typical Premier match consists of 10 players, 5 from each team. However, in some cases, a player may be missing due to the website's inability to access that player's data. This would cause an imbalance between two teams regarding the stat values when in reality, the teams are actually balanced. Therefore, entries like this must be dropped [5].

The following processes are performed to maintain data quality:

1) Cleaning
   Removal of entries with missing/NaN values. Unfortunately, some players' data are not shown properly or are completely missing from the stat page. Therefore, matches with these players must be dropped to reduce noise in data. Matches that end in a draw will also be removed due to several reasons. Firstly, drawn matches are an extremely rare occurrence in Premier matchmaking, only possible when both teams agreed to a draw. Therefore, not only matches that end in a draw could potentially produce excessive noise in the data, drawn matches themselves are the players' choice, not the outcome of their performance.
2) **Aggregation and Conversion**
   Each stat value, from each player from each team, will be compounded into a single column. The average team stat must be obtained first. Then, the averages from each

team will be subtracted, which results in a difference of averages. The difference of averages will be used as the feature columns for the model.

3) **Scaling and Standardization**
Scaling is essential to uphold the model's performance. It establishes equal contribution between features, ensuring that the model is not biased on one feature over another [6]. The method of scaling that will be used is standardization (z-score normalization)—a method that scales the values such that the mean is 0 with a standard deviation of 1. This is a powerful method for models that calculates or utilizes distances between features. Furthermore, standardization is quite robust to outliers as it is not bound to a fixed range.

The resulting dataset will consist of 13 columns, 11 of which are features and 1 label and target column. Every value in each column, except the identity column, will be standardized and scaled according to the process mentioned beforehand.

TABLE I: Dataset Columns

Column Name	Column Type	Description
match_id	identity	identify and differentiate matches
mBias	feature	T-side win-rate of the specific map
CRda	feature	Current Rating—player's current Premier rating
HRda	feature	Highest Rating—player's highest Premier rating all-time
KDda	feature	Kill/Death ratio—ration between player's total kill count and death count
HLTVrda	feature	HLTV rating—player's rating on HLTV
WRda	feature	Win-rate—player's overall win-rate
HSpda	feature	Headshot percentage—how often a player kills an opponent with a headshot
ADRda	feature	Average Damage per Round
CSda	feature	Clutch success—success rate of player winning from a disadvantageous situation
ESda	feature	Entry success—success rate of player entry into the bomb-site
MWRda	feature	Map win-rate—map specific player win-rate
MPRda	feature	Map play-rate—how often the player plays the map
winner	label	binary label—0 if t1 wins else 1

C. Model and Techniques

Due to the erratic nature of CS2 matches, model selection is critical to ensure consistent and desirable output. Considering that all of the datasets are labeled, the type of machine learning model that will be employed is supervised machine learning [7]. Furthermore, the label shown on Table I is categorical, not continuous. Thus, a classification algorithm will be adopted. There are a variety of classification algorithms that are used in machine learning. In this project however, four classification algorithms will be utilized—Support Vector Machines, Logistic Regression, Random Forest, K-Nearest Neighbors [8].

1) Support Vector Machines: A supervised machine learning algorithm that aims to classify data by finding the best line or *hyperplane* that maximizes the margin/distance between two or more classes [9]. The number of features in the training dataset will determine the dimensionality of the hyperplane. SVMs are widely used in machine learning due to its versatility in handling both linear and nonlinear classification tasks.

2) Logistic Regression: A statistical model often used for binary classification, predicting whether event a or b will occur, given a dataset [10]. It predicts the probability that a given input will belong to a specific category. Logistic Regression can be expressed mathematically as:

$$\hat{P} = \frac{e^{b_0+b_1X}}{1+e^{b_0+b_1X}} \tag{1}$$

3) Random Forest: A machine learning algorithm that combines the outputs of multiple decision trees into a single result [11]. Random Forest aims to fix the core problems of a decision tree, overfitting, and bias. By forming an ensemble out of multiple decision trees, the algorithm can predict a more accurate result. Random Forest has three hyperparameters that must be set before model training—-node size, number of trees, and number of sampled features.

4) K-Nearest Neighbors: A supervised machine learning algorithm that classifies data based on proximity and grouping of data points [12]. The goal of a KNN algorithm is to identify a data point's nearest neighbor to assign a label to it. Different types of distance metrics can be utilized by the algorithm depending on the type of data and characteristics.

The library used to create these models is Scikit-learn, a library that provides all the necessary tools to train, test, and evaluate the models. To split the data for training and testing, *the train_test_split* method will be used. Data will be split into a **70:30–Train:Test** ratio. For graphing and visualization purposes, the *matplotlib* library will suffice to draw every necessary graphs and tables.

To further improve the model's performance, the model's *hyperparameters* can be tuned and configured. This is known as *hyperparameter tuning*. Each algorithm has its own distinct hyperparameter, which will affect its performance. The tuning process will be done manually, aiming to maximize the model's overall metric score.

Hyperparameter	Value
kernel	'linear'
c	1.0
gamma	'scale'

(a) SVC

Hyperparameter	Value
solver	'liblinear'
c	1.0

(b) LGR

Hyperparameter	Value
n_estimators	100
min_samples_split	2
min_samples_leaf	1

(a) RF

Hyperparameter	Value
metric	'minkowski'
p	2
n_neighbors	5

(b) KNN

After several testings, the hyperparameter configuration of each model used for the testing results is as shown on the tables above. The configuration has been optimized to maximize the scoring metrics of each model [13].

D. Evaluation Method and Metrics

A machine learning model's performance can be evaluated using evaluation metrics. Most of these metrics are calculated based on true and false predictions, denoted as a *confusion matrix*. Visualizing the confusion matrix is the first step towards evaluating the models, as it serves as a baseline guide on the strengths and weaknesses of a model, allowing the calculations of deeper and more insightful metrics such as *accuracy, precision, recall, F1-score* and *ROC-AUC* [14]. To evaluate the models, the following metrics will be used:

1) Accuracy: Measures proportions of correctly predicted labels out of all the predictions from a sample dataset.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

2) Precision: Measures proportions of true positive predictions with all positive predictions.

$$\text{Precision} = \frac{TP}{TP + FP}$$

3) Recall: Also known as *sensitivity*, is used to measure the proportions of actual positive predictions that were correctly identified by the model.

$$\text{Recall} = \frac{TP}{TP + FN}$$

4) F1-score: The F1-score is the average of both precision and recall, a score that acts as a middle ground between both metrics.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

5) ROC-AUC: The *Receiver Operating Characteristic* (ROC) curve plots the relationship between the model's *True Positive Rate* and its *False Positive rate*. The *Area Under the Curve* (AUC) is the metric used to represent the model's confidence in making predictions correctly [15].

True Positive Rate (TPR):

$$\text{TPR} = \frac{TP}{TP + FN}$$

False Positive Rate (FPR):

$$\text{FPR} = \frac{FP}{FP + TN}$$

[16]

E. Optimizations and Improvements

To enhance and refine the model's performance, further insight on how the model interacts and uses its data must be gained. To help in understanding this, feature importance will be implemented. Feature importance will help to determine the significance of specific features, analyzing which one is the model biased towards [17]. There are a number of ways to calculate feature importance. A method that will be employed is feature importance based on permutation [18]. This method of calculating feature importance works with all types of algorithms. This makes it a perfect candidate, as there are four different algorithms to analyze. The result can then be graphed for a more robust visualization.

By using the information gained from the feature importance, the 'usefulness' of a feature will be revealed. This is helpful to optimize as one way of optimizing the models is by feature selection/pruning. Feature selection is the process of removing irrelevant features from the dataset. This will reduce the dimensionality, simplifying the model in hopes to enhance interpretability and predictive accuracy [19].

IV. Results and Discussion

A. Scores

TABLE II: Model Score Results

Model	accuracy	precision	recall	F1-score	ROC-AUC
Support Vector Classifier	0.9144	0.8876	0.9433	0.9146	0.9152
Logistic Regression	0.9205	0.8982	0.9434	0.9202	0.9211
Random Forest Classifier	0.9144	0.8970	0.9301	0.9136	0.9148
KNeighbors Classifier	0.8899	0.8639	0.9182	0.8902	0.8907

From the results in the Table II, it is reasonable to infer that all models perform relatively well, averaging a score of around 90% accuracy. Logistic regression edges ahead of SVC and Random Forest slightly with a margin of less than 1%. SVC and Random Forest performs rather similarly for their accuracy, F1-score, and ROC-AUC. From the test results, the SVC model seems to have a lower precision score compared to Random Forest. On the other hand, Random Forest has a lower recall score compared to SVC, meaning its outputs more false negatives. The KNeighbors Classifier falls slightly behind the other models at every metric. This is due to the fact that KNN tends to perform poorly compared to other algorithms with data of dimensionality [20].

B. Receiver Operating Characteristics - ROC

As shown in Fig. 2, the models have a high discriminatory power. All four curves shows similar characteristics—steep rise towards the top left of the chart and flattens along the top. This observation is supported by the results in Table II, which show all four models having a high ROC-AUC score. Similar to previous results, the KNN model falls slightly short compared to the rest. Its curve is noticeably lower with an AUC score of 0.89—. Others score above 0.90.

C. Feature Importance

The graph shown in Fig. 1 shows that there are 2 particularly dominant features—MWRda (Map Win-rate) and CRda (Current Rating), both contributing over 50% of the model predictions out of 12 total features. This implies that the player's overall win-rate on the match's map, as well as their current Premier rating, affects the chances of them winning/losing significantly. It is important to note that the feature importances are relatively consistent over all four models. KNN however, has an interestingly more balanced

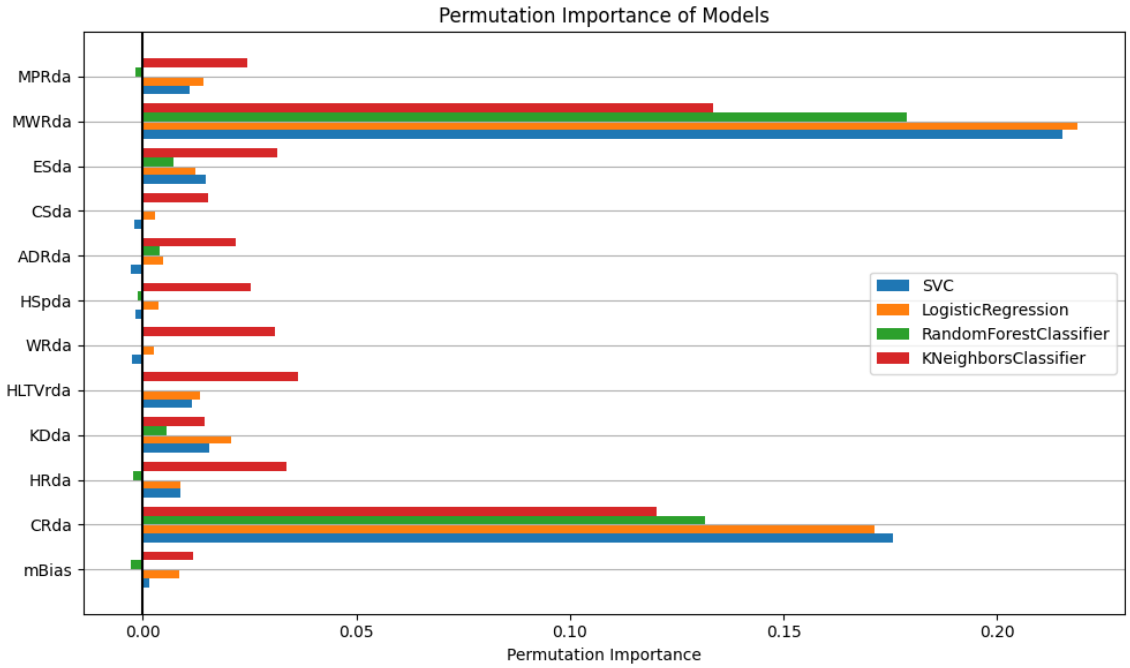

Fig. 1: Feature Importance based on Permutation

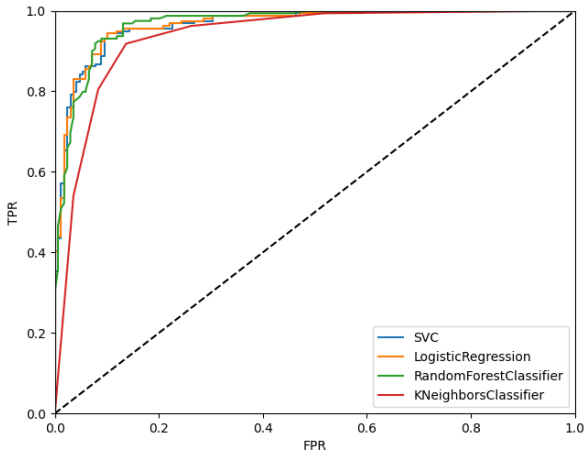

Fig. 2: ROC Curve

feature importance distribution—higher lows and lower highs. It can be inferred that other features contribute more on KNN's score compared to the other models.

D. Feature Selection and Optimized Results

To optimize the model, some features will be removed from the training dataset. According to Fig. 1, the features with the least importance (not in order) are: mBias, WRda, HSpda, ADRda, CSda, and MPRda. These features will be dropped and the new feature-pruned dataset will be fed into the algorithms again.

TABLE III: Model Score Results Post-Feature Pruning

Model	accuracy	precision	recall	F1-score	ROC-AUC
Support Vector Classifier	0.9144	0.8876	0.9434	0.9146	0.9152
Logistic Regression	0.9144	0.8876	0.9434	0.9146	0.9152
Random Forest Classifier	0.9083	0.8910	0.9245	0.9074	0.9087
KNeighbors Classifier	0.9205	0.9182	0.9182	0.9182	0.9204

Although it seems minuscule, the KNN algorithm improved by a considerable margin. After the unnecessary features were removed, it improves its accuracy score from 89% to 92%, on par with Logistic Regression. However, it is important to note that while the feature selection process is beneficial for the KNN's performance, the same cannot be said for the other models. For example, Logistic Regression drops from 92& accuracy to 91%. This is likely due to other models requiring such features; however, it's irrelevant as it is to create correlations between one another. Additionally, the Random Forest model also drops slight in terms of its overall score—from 91.44% accuracy to 90.88%.

V. Conclusion

A. Summary

In this study, we demonstrated the feasibility of using machine learning to predict the outcome of CS2 matches.The models developed in this study achieved consistent predictive accuracy. Of the four models tested, the Logistic Regression model was particularly effective, followed by Support Vector Machines and Random Forest as second and third, respectively, and the K-Nearest Neighbors model performing the worst overall. However, after feature selection and optimization, the KNN model improved significantly, achieving results comparable to the other 3 models. We find that specific

features such as the map win rate (MWRda) and current rating (CRda) greatly influence the accuracy of the prediction models.

B. Suggestions and Future Works

One of the main bottlenecks during the development of the project is during data collection. As mentioned, the scraping library used is Selenium. However, we realized that other libraries such as BeautifulSoup4 and Requests could achieve the same thing, but faster and more efficient. This is because Selenium requires an actual web driver to navigate the pages, whilst BS4 and HTML Requests return only the necessary HTML information.

Future works will focus on exploring other possible models, as well as refining the current model's performance. Future studies may also explore similar models for other CS2 game modes such as '*Official* Competitive matches. We hope that this paper could give insight to CS2 players regarding the factors in winning a match as well as a stepping stone for future works and papers to improve upon.

SUPPLEMENTARY CODES

All the codes used in this paper can be accessed through the following link: Click here to access the codes.

CREDIT AUTHORSHIP CONTRIBUTION STATEMENT

Benedictus Bellion Leander, Christian Antonio Yotanka Tampubolon, Christopher Darren Muljono: Conceptualization, Data Curation, Formal analysis, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing - original draft, Writing - review & editing. **Nunung Nurul Qomariyah, Raymond Bahana:** Conceptualization, Writing - Review & Editing, Supervision, Project administration, Funding acquisition.

REFERENCES

[1] B. S. Choi, L. K. Foo, and S.-L. Chua, "Predicting football match outcomes with machine learning approaches," in *MENDEL*, vol. 29, no. 2, 2023, pp. 229–236.

[2] T. Horvat, J. Job, R. Logozar, and Č. Livada, "A data-driven machine learning algorithm for predicting the outcomes of nba games," *Symmetry*, vol. 15, no. 4, p. 798, 2023.

[3] L. Lin, "League of legends match outcome prediction," *Comput. Sci. Dept., Univ. Stanford, Stanford, CA, USA, Rep*, 2016.

[4] H.-H. Xiao, W.-K. Yang, J. Hu, Y.-P. Zhang, L.-J. Jing, and Z.-Y. Chen, "Significance and methodology: Preprocessing the big data for machine learning on tbm performance," *Underground Space*, vol. 7, no. 4, pp. 680–701, 2022.

[5] S. Rao, P. Poojary, J. Somaiya, and P. Mahajan, "A comparative study between various preprocessing techniques for machine learning," *Int. J. Eng. Appl. Sci. Technol*, vol. 5, no. 3, pp. 2455–2143, 2020.

[6] M. M. Ahsan, M. P. Mahmud, P. K. Saha, K. D. Gupta, and Z. Siddique, "Effect of data scaling methods on machine learning algorithms and model performance," *Technologies*, vol. 9, no. 3, p. 52, 2021.

[7] C. Donalek, "Supervised and unsupervised learning," in *Astronomy Colloquia. USA*, vol. 27, 2011, p. 8.

[8] P. C. Sen, M. Hajra, and M. Ghosh, "Supervised classification algorithms in machine learning: A survey and review," in *Emerging Technology in Modelling and Graphics: Proceedings of IEM Graph 2018*. Springer, 2020, pp. 99–111.

[9] S. Suthaharan, "Support vector machine," in *Machine Learning Models and Algorithms for Big Data Classification*. SpringerLink, 1970.

[10] T. G. Nick and K. M. Campbell, "Logistic regression," *Topics in biostatistics*, pp. 273–301, 2007.

[11] S. J. Rigatti, "Random forest," *Journal of Insurance Medicine*, vol. 47, no. 1, pp. 31–39, 2017.

[12] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, no. 1, pp. 381–386, 2020.

[13] M. R. Hossain and D. Timmer, "Machine learning model optimization with hyper parameter tuning approach," *Glob. J. Comput. Sci. Technol. D Neural Artif. Intell*, vol. 21, no. 2, p. 31, 2021.

[14] Ž. Vujović *et al.*, "Classification model evaluation metrics," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 6, pp. 599–606, 2021.

[15] K. H. Zou, A. J. O'Malley, and L. Mauri, "Receiver-operating characteristic analysis for evaluating diagnostic tests and predictive models," *Circulation*, vol. 115, no. 5, pp. 654–657, 2007.

[16] M. Hossin and M. N. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International journal of data mining & knowledge management process*, vol. 5, no. 2, p. 1, 2015.

[17] A. Zien, N. Krämer, S. Sonnenburg, and G. Rätsch, "The feature importance ranking measure," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2009, Bled, Slovenia, September 7-11, 2009, Proceedings, Part II 20*. Springer, 2009, pp. 694–709.

[18] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: a corrected feature importance measure," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, 2010.

[19] M. Dash and H. Liu, "Feature selection for classification," *Intelligent data analysis*, vol. 1, no. 1-4, pp. 131–156, 1997.

[20] N. Kouiroukidis and G. Evangelidis, "The effects of dimensionality curse in high dimensional knn search," in *2011 15th Panhellenic Conference on Informatics*. IEEE, 2011, pp. 41–45.