International Journal of Scientific Research in Computer Science and Engineering
Vol.**12**, Issue.**6**, pp.**34-40**, December **2024**
E-ISSN: 2320-7639
Available online at: www.isroset.org

Research Article

Real-time Ad Click Prediction: Logistic Regression vs. Decision Tree

Divyanshu Prajapati[1], Indranil Samanta[2], Amit Kumar[3], Pankaj Dumka[4*]

[1,2,3]Computer Science and Engineering, Jaypee University of Engineering and Technology, Guna, India
[4]Mechanical Engineering Department, Jaypee University of Engineering and Technology, Guna, India

**Corresponding Author: p.dumka.ipec@gmail.com*

Received: 25/Oct/2024; **Accepted**: 27/Nov/2024; **Published**: 31/Dec/2024

Abstract— This study studies the usefulness of different algorithms based on machine learning viz. Logistic Regression and Decision Tree, in forecasting the real-time ad clicks. Python has been used to implement and evaluate these models on a static dataset. Compared to Logistic Regression, the Decision Tree model has established better performance i.e. attaining an F1-score of 0.744 versus 0.7154. However, the Decision Tree's performance might be affected by overfitting and dataset characteristics. Future research should explore joint methods, deep learning, addressing imbalanced data, and privacy-preserving techniques to improve ad click prediction models.

Keywords— Ad click prediction; Logistic regression; Linear regression; Decision tree; Python programming

1. Introduction

Ad click prediction has become a decisive viewpoint of digital marketing, enabling businesses to optimize ad campaigns, allocate budgets effectively, and enhance user experience [1]. Accurately predicting the user behaviour can enables advertisers to present aimed messages to the suitable audience at optimum times [2], [3]. This research investigates the effectiveness of Logistic Regression and Decision Tree models in predicting the ad click-through rates [2], [4].
Accurate ad click prediction poses many benefits to both advertisers and users [5], [6], such as:

- Improved Targeting: By detecting potential clickers, advertisers can give resources to the most promising campaigns, maximizing return on investment (ROI).
- Enhanced User Experience: Personalized advertising can reduce the ad fatigue and increases the users satisfaction by delivering relevant content.
- Optimized Ad Spend: By avoiding extravagant ad impressions, businesses can distribute budgets more efficiently.

Numerous studies have searched the application of ML algorithms in ad click prediction [7], [8]. For example, the Criteo dataset has been widely used to the benchmark machine learning models for the ad click prediction due to its large scale and various features. Another example is the Avazu dataset that includes categorical and numerical features, thus allowing the researchers to evaluate the effectiveness of feature engineering and preprocessing methods [9]. These datasets have shown the ability of Logistic Regression to handle the high-dimensional data efficiently and also the flexibility of Decision Trees in capturing non-linear interactions between the features. For example, researchers have employed Logistic Regression for its simplicity and interpretability in predicting user behaviour based on demographic and engagement features [7]. Decision Trees, on the other hand, have been favoured for their ability to capture complex patterns in data [10].

Early approaches to ad click prediction were based on the simple statistical models like Logistic Regression. However, with the arrival of machine learning the more sophisticated techniques have been developed.

ANN (artificial neural network) architectures like Convolutional Neural Networks (CNNs) [11] and Recurrent Neural Networks (RNNs) [11], [12] have been employed effectively in similar domains, leveraging their ability to handle high-dimensional and successive data, respectively [13]. For illustration, studies utilizing Long Short-Term Memory (LSTM) networks have proved improved accuracy in predicting user visit patterns by capturing time-based needs [14]. Case studies involving companies like Alibaba and Google have also showcased the power of deep learning in large-scale ad click prediction systems thereby highlighting the capacity for scalability and increased precision in managing diverse user data [15]. Although, these models are computationally intensive but can capture intricate data patterns, outperforming traditional ML algorithms in large-scale datasets [16] very efficiently. However, their application remains limited due to challenges in interpretability and data privacy concerns. This paper builds

upon existing research by giving a comparative analysis of Logistic Regression and Decision Tree models, emphasizing their strengths and weaknesses in the context of ad click prediction.

Decision Trees in particular uses a non-linear framework to model complex dependencies between variables and the target variable [17]. Now a days ensemble techniques, such as, Gradient Boosting and Random Forest have developed as a popular choices due to their ability to enhance projecting accuracy by combining multiple models. Deep learning, specifically neural networks, has demonstrated significant potential in processing extensive datasets and finding complex patterns in user behaviour [18].

Python, with its vast environment of modules such as scikit-learn [19], pandas [20], [21], and NumPy [22], [23], [24], [25], has become a foundation for machine learning research and applications. Its ease and flexibility enable quick prototyping and deployment of models which make it an ideal choice for the tasks like ad click prediction [26], [27], [28], [29], [30]. Moreover, Python's ability to integrate effortlessly with visualization libraries and deployment frameworks further improves its appeal in both academic and industrial settings. Through a static dataset, their capabilities in forecasting real-time ad clicks are compared based on metrics such as F1-score, Mean Squared Error (MSE) [11], [31], [32], and Area Under the Receiver Operating Characteristic Curve (AUC) [33]. The study also examines the questions associated with these models, including overfitting and dataset characteristics, while offering recommendations for future enhancements.
Following is the Python's value when it comes to ad click prediction:

- Data Preprocessing: Python enables efficient data cleaning, controlling missing values, and attribute engineering.
- Model Selection and Training: A vast selection of machine learning algorithms is accessible within Python libraries, facilitating experimentation and optimization.
- Model Evaluation: Python offers several metrics, including accuracy, recall, and F1-score, to measure model performance.
- Real-time Implementation: Libraries like TensorFlow and PyTorch [34] facilitate the deployment of models based on deep learning for real-instantaneous predictions.

While numerous studies have explored ad click prediction, several challenges remain, such as:

- Many models struggle to handle the rapid influx of data required for real-time decision-making.
- The imbalance between clicked and non-clicked ads can bias model training.
- Some models, particularly deep learning models, are naturally difficult to interpret, making it hard to realize their decision-making processes.

To address these challenges, this research will compare the performance of Logistic Regression and Decision Tree models on a static Kaggle dataset. We will then explore the potential of Decision Trees for real-time ad click prediction, leveraging Python's capabilities to handle dynamic data and adapt to changing user behaviour.

2. Methodology

The dataset included the number of hours one spends online per day, age, income, the usage of the internet, gender, and the fact of whether or not the user clicked on the ad to predict whether ads are clicked or not. City names, description of ads, and timestamps were deleted from the dataset. So, in the cleaned data set, there was no missing value, and it was ready for training. The target variable is Clicked on Ad, representing whether a user clicked on the advertisement (1) or not (0) [35]. Data preprocessing steps included handling missing values, encoding categorical variables, and splitting the data into training and testing sets. Handling missing values was crucial to ensure that the model did not encounter undefined or biased results due to incomplete data. Encoding categorical variables transformed non-numeric data into numerical formats, enabling the machine learning algorithms to process these features effectively [36]. Furthermore, feature scaling techniques such as normalization and standardization were employed to ensure that all numerical features contributed equally to the model's learning process. The importance of these preprocessing steps lies in their ability to enhance model accuracy and generalization, especially when dealing with high-dimensional or noisy datasets [37].

This is further divided into two subcategories: the training set of 13,325 samples and the testing set of 3,332 samples. The three models selected were Linear Regression [30] and Logistic Regression [38] and Decision Tree Regression [39]. An application of an F1 score is given in the evaluation of the degree to which the selected models performed well [40]. A line will rather be favoured than logistics for static data sets as this would make quite a better prediction for the outcome of the binary. Real-time or dynamic data, which is the case for Decision Tree Regression, would, therefore, be the best choice for flexibility and efficiency. Hyperparameter tuning was supervised for Decision Tree model to optimize its performance [41]. Specific hyperparameters tuned included the maximum depth of tree, the minimum samples required to split a node, and the criterion for measuring the quality of a split (e.g., Gini impurity or entropy). The ranges look at for these parameters were systematically varied: maximum depth ranged from 3 to 20, the minimum samples split ranged from 2 to 10, and both Gini and entropy criteria were evaluated [42]. Grid Search Cross-Validation was used as the optimization methodology, enabling the recognition of the optimal hyperparameter arrangement based on performance metrics such as F1-score and Mean Squared Error (MSE). This systematic approach guaranteed a balance between model complexity and generalization thereby avoiding the overfitting and maximizing predictive accuracy.

2.1 Linear Regression:

It is a statistical technique that demonstrates the correlation between a dependent varaible and one or more independent variables. In the context of ad click prediction, it can estimate the chance of a user clicking on an ad based on factors like user demographics, ad characteristics, and contextual information. The mathematical equation for simple linear regression is [30], [43]:

$$y = mx + b \quad (1)$$

For multiple linear regression, the equation becomes:

$$y = b_0 + b_1 \times x_1 + b_2 \times x_2 + \ldots + b_n \times x_n \quad (2)$$

2.2 Logistic Regression [44], [45]:

It is a stochastic method used to show the probability of a binary outcome (like whether a user will click on an ad) given one or more predictor variables. It is a popular choice for ad click prediction due to its suitability for binary outcomes and its interpretability. Its mathematical foundation involves the logistic function, that maps any real number between 0 and 1, as shown in Eqn. 3.

$$P(y = 1|x) = 1 / (1 + exp(-z)) \quad (3)$$

Where, $P(y = 1|x)$ is the probability associated with the positive outcome (e.g., click) given the input features x and z is a linear combination of the input features and their coefficients (Eqn. 2).

2.3 Decision Tree Regression :

It is an algorithm which is based on supervised learning that can be used for both classification and regression. In the context of ad click prediction, decision trees can be used to predict the continuous numerical value of a user's click probability based on various factors [39]. It works by recursively splitting the data into smaller subsets which are based on specific conditions. Each internal node signifies a test on an attribute, and each branch denotes a possible test outcome [46]. The leaf nodes represent the final prediction. For regression tasks, the prediction at a leaf node is typically the average value of the target variable (click probability) for the training data that falls into that leaf.

While decision trees don't have a straightforward mathematical equation like linear regression, their decision-making process can be represented as a series of conditional statements. For instance, a simple decision tree might look like as shown below:

```
If Age < 30:
    If Income > 50,000:
        Predict Click Probability = 0.8
    Else:
        Predict Click Probability = 0.3
Else:
    If Gender = Male:
        Predict Click Probability = 0.6
    Else:
        Predict Click Probability = 0.4
```

3. Python Implementation

In this section a detailed explanation of how the python functions has been used and what are they doing is explained is a great detail.

3.1 Linear Regression:

```
from sklearn.linear_model import
LinearRegression

model_lr = LinearRegression()
model_lr.fit(X_train, y_train)
```

This code snippet focuses on creating and training a linear regression model. Here's a breakdown:

- **Importing Linear Regression**
  First the LinearRegression class is imported from the sklearn.linear_model module which will help in the creation of a linear regression model.
- **Creating a Model**
  Then an instance of the LinearRegression class has been made and assigns it to model_lr.
- **Fitting the Model**
  Then the developed linear regression model has been instructed on the training data (X_train and y_train). The model learns the coefficient that best fits the linear relationship between the features the target variable.

```
from sklearn.metrics import f1_score

y_pred_lr = model_lr.predict(X_test)
f1_lr = f1_score(y_test,
y_pred_lr.round())
```

This code fragment focuses on evaluating the performance of a logistic regression model (model_lr) after making predictions on a test set (X_test). Here's a breakdown:

- **Importing f1_score**
  First the f1_score function has been imported from the sklearn.metrics module. F1-score is a system of measurement that ties precision and recall.
- **Predicting with model_lr**
  Then the trained model (model_lr) has been used to create predictions on the X_test data. The expected probabilities are stored in y_pred_lr.
- **Calculating F1-Score**
  Finally the F1-score has been judged by comparing the true labels (y_test) with the rounded predicted probabilities (y_pred_lr.round()). Rounding is necessary because the F1-score is typically used for binary classification, where predictions are either 0 or 1.

3.2 Logistic Regression:

```
from sklearn.linear_model import
LogisticRegression

model_logistic = LogisticRegression()
model_logistic.fit(X_train, y_train)
```

This code snippet focuses on creating and training a logistic regression model. Here's a breakdown:

- **Importing LogisticRegression**
  To begin, the LogisticRegression class is imported from the sklearn.linear_model module.
- **Creating a Model**
  Then an instance of the LogisticRegression class has been made and assigns to the variable model_logistic.
- **Fitting the Model**
  Finally the model is trained on the training data (X_train and y_train). The model learns the coefficients that are best to predict the probability of the positive class (e.g., click) based on the features in X_train.

```
y_pred_logistic =
model_logistic.predict(X_test)
f1_logistic = f1_score(y_test,
y_pred_logistic)
```

- **Making Predictions**
  First the trained logistic regression model (model_logistic) is used to make guesses on the X_test data. The predicted class labels (0 or 1) are stored in y_pred_logistic.
- **Calculating F1-Score**
  Subsequently, the F1-score is figured by assessing the actual labels (y_test) with the predicted labels (y_pred_logistic).

3.3 Decision Tree Regression:

```
from sklearn.tree import
DecisionTreeRegressor

model_dt = DecisionTreeRegressor()
model_dt.fit(X_train, y_train)
```

This code snippet focuses on creating and training a decision tree regression model. Here's a breakdown:

- **Importing DecisionTreeRegressor:**
  Initially, the DecisionTreeRegressor class is imported from the sklearn.tree module. This class is subsequently utilized to construct a decision tree regression model.
- **Creating a Model:**
  Then an instance of the DecisionTreeRegressor class is built and assigns it to the variable model_dt.
- **Fitting the Model:**
  Next, the model is trained on the training data (X_train and y_train). The model learns decision rules that optimally predict the continuous numerical target variable based on the features in the training data.

```
y_pred_dt = model_dt.predict(X_test)
f1_dt = f1_score(y_test,
y_pred_dt.round())
```

This code snippet focuses on making predictions with the trained decision tree regression model and evaluating its performance using the F1-score. However, there's a slight issue with using the F1-score for regression tasks.

- **Making Predictions:**
  To begin with, the trained decision tree regression model (model_dt) is used to do some predictions on the X_test data. The resulting predicted continuous numerical values are then stored in y_pred_dt.
- **Calculating F1-Score:**
  Then F1-score has been calculated by judging the true labels (y_test) with the rounded guessed values (y_pred_dt.round()). However, rounding the predicted values for a regression task is not appropriate. The F1-score is typically used for binary classification problems.

4. Results and Discussion

The plot of Actual vs Predicted Values for Ad Clicks for linear regression case has been shown in the Figure 1, which provides a representation of how well a linear regression model is predicting ad clicks. The blue line represents the actual click values (0 or 1), while the red line represents the predicted values. Ideally, the two lines should be closely aligned, indicating accurate predictions. However, in this case, the predicted values seem to be concentrated around 0.1 and 1.0, while the actual values are mostly 0 or 1. This indicates that the model is unable to differentiate effectively between clicks and non-clicks at a granular level.

The F1-score of 0.71283 implies that the model performs reasonably well; nevertheless, it is important to consider the context. In ad click prediction, where the dataset is often imbalanced (with more non-clicks than clicks), a high F1-score may not necessarily reflect the model's ability to accurately predict positive cases (clicks). Therefore, while the F1-score is a useful metric, it is essential to examine other metrics like precision, recall, and confusion matrices to get a more thorough evaluation of the model's performance.

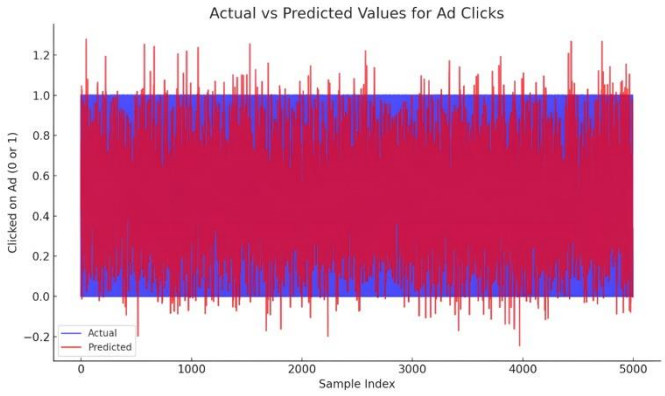

Figure 1: Actual vs Predicted value (linear regression)

Figure 2 shows the Receiver Operating Characteristic (ROC) curve for the developed logistic regression model.

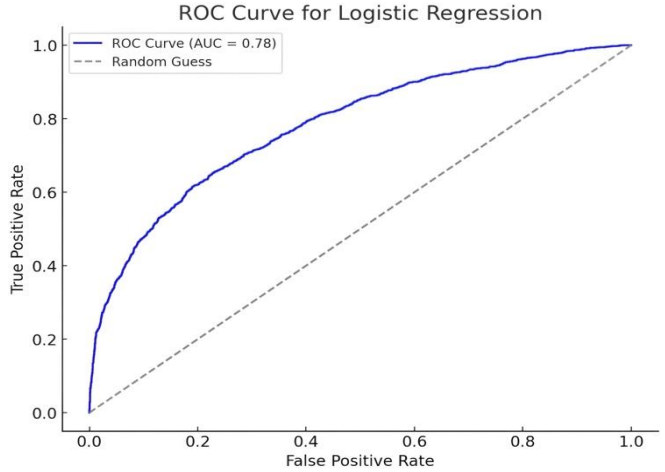

Figure 2: ROC Curve (logistic regression)

The ROC curve is a powerful tool for picturing the working of a binary classification. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold sets. The TPR measures the amount of actual positive cases correctly identified as positive, while the FPR measures the ratio of actual negative cases incorrectly identified as positive. The diagonal line (shown in blue) represents a random classifier, which would have an equal TPR and FPR at all points. An ideal classifier should have a curve that is as far away from this random line as possible, towards the top-left corner. The blue curve in the graph represents the ROC curve for the logistic regression model.
As observed in Fig. 2, the ACU (area under the curve) is 0.78. This number shows the overall performance of the model. An AUC of 0.5 would be comparable to random guessing, while an AUC of 1.0 means a perfect classifier. For logistic regression, an F1-score of 0.7154 implies that the model demonstrates decent performance in terms of both precision and recall. It is a good balance between the two metrics.

Figure 3 visualizes the relationship between the actual ad click values (blue dots) and the predicted values (red dots) obtained from a decision tree regression model. The x-axis represents the "Daily Time Spent on Website", while the y-axis represents the "Clicked on Ad" variable. It is observed that the actual values tend to cluster around 0 and 1, indicating that the ad clicks are binary (either clicked or not clicked). The predicted values, represented by the red dots, are scattered across the y-axis. This implies that the decision tree model is not effectively describing the underlying patterns in the data. The model might be overfitting of the training data, thus leading to a poor generalization on the test data. A Mean Squared Error (MSE) of 0.2349 is obtained for this case, showing that the model's predictions are not very accurate. The F1-score for this model is 0.744.

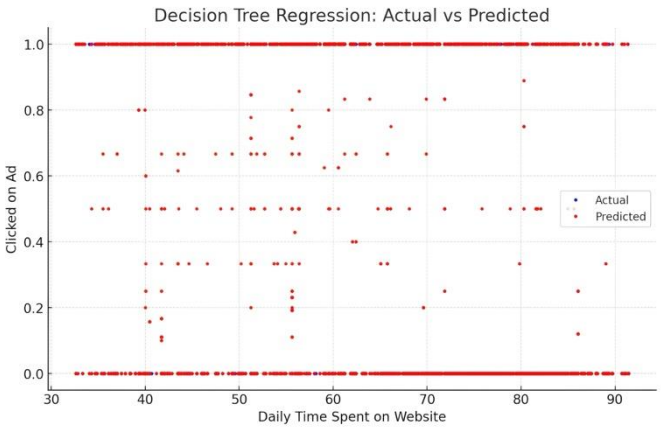

Figure 3: Decision Tree Regression: Actual vs Predicted

Table 1: Time spent

Daily Time Spent	62.3	41.7	44.4	59.9
Age	32	31	30	28
Area Income	69481.9	61840.3	57877.2	56180.9
Daily Internet Usage	172.8	207.2	172.8	207.2
Ad Topic Line	Decentralized real-time circuit	Optional full-range projection	Total 5th generation standardization	Balanced empowering success
City	Lisafort	West Angelabury	Reyesfurt	New Michael
Male	1	1	0	0
Country	Svalbard & Jan Mayen Islands	Singapore	Guadeloupe	Zambia
Clicked on Ad	0	0	0	0

This Table 1 includes time spent on the site, age, area income, internet usage, gender, and the city. The model largely relies on Time Spent on Site (how engaged users are), Gender-which impacts the behaviours based upon demographics-and Daily Internet Usage, which determines how familiar someone is with the internet; this decides his interaction probability with the ad.

6. Conclusion and Future Scope

This study investigated the efficiency of Logistic Regression and Decision Tree models for real-time ad click prediction. The Decision Tree model surpassed Logistic Regression in

performance metrics, achieving an F1-score of 0.744 compared to 0.7154. However, its susceptibility to overfitting and dependence on dataset characteristics highlight the need for further improvements.

Future research should explore ensemble methods, such as Random Forests and Gradient Boosting, to enhance prediction accuracy and robustness. These methods could address current limitations by combining multiple decision trees to reduce variance and prevent overfitting. For example, studies have shown that Random Forests excel in handling high-dimensional datasets and missing values, while Gradient Boosting effectively models complex patterns in data by sequentially optimizing weak learners. Incorporating such methodologies could offer a more reliable and scalable solution for ad click prediction, drawing insights from existing research on ensemble learning applications in similar domains. Additionally, deep learning models, though computationally demanding, hold promise for capturing complex data patterns. Addressing imbalanced datasets, incorporating privacy-preserving techniques, and integrating real-time data streams are other potential areas for future work. These advancements could significantly improve the reliability and scalability of ad click prediction systems.

Data Availability
Data will be made available on request.

Conflict of Interest
We do not have any conflict of interest.

Funding Source
None

Authors' Contributions
Authors 1 conceived the study and performed conducive literary research, review, perceiving the conceptual analysis taken up. Author 2 contributed to the code and conceptualized the execution process. Author 3 processed the data and played an instrumental role in the literature review. Author 4 directed the progress of the study and reviewed and finalized the article.

Acknowledgements
The authors are grateful to Dr. Dhananjay R. Mishra for his guidance during the preparation of this manuscript.

References

[1] H. B. Mcmahan *et al.*, "Ad click prediction: A view from the trenches," *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, vol. Part F1288, pp.**1222–1230, 2013**, doi: 10.1145/2487575.2488200.

[2] S. Saraswathi, V. Krishnamurthy, D. Venkata Vara Prasad, R. K. Tarun, S. Abhinav, and D. Rushitaa, "Machine learning based ad-click prediction system," *Int. J. Eng. Adv. Technol.*, Vol.**8**, Issue.**6**, pp.**3646–3648, 2019**, doi: 10.35940/ijeat.F9366.088619.

[3] A. Clemenshia and V. M., "Click Through Rate Prediction for Display Advertisement," *Int. J. Comput. Appl.*, Vol.**136**, No.1, pp.**18–24, 2016**, doi: 10.5120/ijca2016908332.

[4] R. Kumar, S. M. Naik, V. D. Naik, S. Shiralli, V. G. Sunil, and M. Husain, "Predicting clicks: CTR estimation of advertisements using Logistic Regression classifier," in *Souvenir of the 2015 IEEE International Advance Computing Conference, IACC 2015*, pp.**1134–1138, 2015.** doi: 10.1109/IADCC.2015.7154880.

[5] O. S. Bratus and P. I. Bidyuk, "Towards Click-Through Rate Prediction in Online Advertising," *Probl. Appl. Math. Math. Model.*, pp.**3–17, 2024**, doi: 10.15421/322301.

[6] Y. Yang and P. Zhai, "Click-through rate prediction in online advertising: A literature review," *Inf. Process. Manag.*, vol. 59, no. 2, p. **102853, 2022**, doi: https://doi.org/10.1016/j.ipm.2021.102853.

[7] M. Kamal and T. A. Bablu, "International Journal of Applied Machine Learning and Computational Intelligence Machine Learning Models for Predicting Click-through Rates on social media: Factors and Performance Analysis," *Int. J. Appl. Mach. Learn. Comput. Intell.*, vol. 4, no. 11, pp. **1–14, 2022.**

[8] A. Shah and S. Nasnodkar, "The Impacts of User Experience Metrics on Click-Through Rate (CTR) in Digital Advertising: A Machine Learning Approach," *Sage Sci. Rev. Appl. Mach. Learn.* , vol. 4, no. 1, pp. **27–44, 2021**.

[9] E. Bicici, "Instance Weighting in Neural Networks for Click-Through Rate Prediction," in *2023 Innovations in Intelligent Systems and Applications Conference, ASYU 2023*, pp.**1–5, 2023.** doi: 10.1109/ASYU58738.2023.10296657.

[10] G. De Ath and K. E. Fabricius, "Death Fabricius_2000 _CART," *Ecology*, vol. 81, no. 11. pp. 3178–3192, 2000.

[11] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural network design*. PWS Publishing Co., 1997.

[12] P. Dhruv and S. Naskar, "Image Classification Using Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN): A Review," in *Machine Learning and Information Processing*, D. Swain, P. K. Pattnaik, and P. K. Gupta, Eds., Singapore: Springer Singapore, 2020, pp. 367–381.

[13] I. Banerjee *et al.*, "Comparative effectiveness of convolutional neural network (CNN) and recurrent neural network (RNN) architectures for radiology text report classification," *Artif. Intell. Med.*, vol. 97, pp. 79–88, 2019, doi: https://doi.org/10.1016/j.artmed.2018.11.004.

[14] C. Sampaio Descovi, A. Carlos Zuffo, S. Mohammadizadeh, L. F. Murillo Bermúdez, and D. Alfonso Sierra, "Utilizing Long Short-Term Memory (Lstm) Networks for River Flow Prediction in the Brazilian Pantanal Basin," *Holos*, vol. 5, no. 39, pp. 1–16, 2023, doi: 10.15628/holos.2023.16315.

[15] L. Micol Policarpo *et al.*, "Machine learning through the lens of e-commerce initiatives: An up-to-date systematic literature review," *Comput. Sci. Rev.*, vol. 41, p. 100414, 2021, doi: https://doi.org/10.1016/j.cosrev.2021.100414.

[16] L. E. Lwakatare, A. Raj, I. Crnkovic, J. Bosch, and H. H. Olsson, "Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions," *Inf. Softw. Technol.*, vol. 127, p. 106368, 2020, doi: https://doi.org/10.1016/j.infsof.2020.106368.

[17] R. Wang, G. Fu, B. Fu, and M. Wang, "Deep & cross network for ad click predictions," *2017 AdKDD TargetAd - conjunction with 23rd ACM SIGKDD Conf. Knowl. Discov. Data Mining, KDD 2017*, 2017, doi: 10.1145/3124749.3124754.

[18] J. Ren, J. Zhang, and J. Liang, "Feature engineering of click-through-rate prediction for advertising," in *Lecture Notes in Electrical Engineering*, Q. Liang, X. Liu, Z. Na, W. Wang, J. Mu, and B. Zhang, Eds., Singapore: Springer Singapore, 2020, pp. 204–211. doi: 10.1007/978-981-13-6508-9_26.

[19] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.," 2022.

[20] P. Dumka *et al.*, "Development and implementation of a Python functions for automated chemical reaction balancing," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 34, no. 3, pp. **1557–1565, 2024,** doi: 10.11591/ijeecs.v34.i3.pp1557-1565.

[21] W. McKinney, *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.," 2012.

[22] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: A structure for efficient numerical computation," *Comput. Sci. Eng.*, vol. 13, no. 2, pp. **22–30, 2011**, doi: 10.1109/MCSE.2011.37.

[23] Y. Raghuvanshi, D. R. Mishra, and P. Dumka, "Understanding Sets with the help of Python 1," *Int. J. Nov. Res. Dev.*, vol. 7, no. 10, pp. 136–142, 2022.

[24] A. R. Joshi, A. Deo, A. Parashar, D. R. Mishra, and P. Dumka, "Modelling Steam Power Cycle using Python," *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 9, no. 1, pp. 152–162, 2023.

[25] P. Dumka, N. Samaiya, S. Gandhi, and D. R. Mishra, "Modelling of Hardy Cross Method for Pipe Networks," *Int. J. Mech. Eng.*, vol. **10**, no. 2, pp. **1–8, 2023.**

[26] J. W. B. Lin, "Why python is the next wave in earth sciences

computing," *Bull. Am. Meteorol. Soc.*, Vol.**93**, no.**12**, pp.**1823–1824, 2012**, doi: 10.1175/BAMS-D-12-00148.1.
[27]D. Nofriansyah and H. Freizello, "Python Application: Visual Approach of Hopfield Discrete Method for Hiragana Images Recognition," *Bull. Electr. Eng. Informatics*, Vol.7, no.4, pp.**609-614, 2018**, doi: 10.11591/eei.v7i4.691.
[28]M. F. Sanner, "Python: A programming language for software integration and development," *J. Mol. Graph. Model.*, vol. 17, no. 1, pp. **57–61, 1999.**
[29]P. S. Pawar, D. R. Mishra, and P. Dumka, "Solving First Order Ordinary Differential Equations using Least Square Method : A comparative study," *Int. J. Innov. Sci. Res. Technol.*, vol. 7, no. 3, pp. **857–864, 2022.**
[30]P. Dumka, R. Dumka, and D. R. Mishra, *Numerical Methods Using Python*. BlueRose, **2022.**
[31]P. Dumka, R. Chauhan, and D. R. Mishra, "Experimental and theoretical evaluation of a conventional solar still augmented with jute covered plastic balls," *J. Energy Storage*, vol. 32, no. June, p. **101874, 2020**, doi: 10.1016/j.est.2020.101874.
[32]R. Chauhan, P. Dumka, and D. R. Mishra, "Modelling conventional and solar earth still by using the LM algorithm-based artificial neural network," *Int. J. Ambient Energy*, vol. 43, no. 1, pp. 1389–1396, 2022, doi: 10.1080/01430750.2019.1707113.
[33]J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 3, pp. 299–310, 2005, doi: 10.1109/TKDE.2005.50.
[34]S. Imambi, K. B. Prakash, and G. R. Kanagachidambaresan, "PyTorch," in *EAI/Springer Innovations in Communication and Computing*, K. B. Prakash and G. R. Kanagachidambaresan, Eds., Cham: Springer International Publishing, 2021, pp. 87–104. doi: 10.1007/978-3-030-57077-4_10.
[35]V. Singh, B. Nanavati, A. K. Kar, and A. Gupta, "How to Maximize Clicks for Display Advertisement in Digital Marketing? A Reinforcement Learning Approach," *Inf. Syst. Front.*, vol. 25, no. 4, pp. 1621–1638, 2023, doi: 10.1007/s10796-022-10314-0.
[36]C. Yu, Z. Zhang, C. Lin, and Y. J. Wu, "Can data-driven precision marketing promote user ad clicks? Evidence from advertising in WeChat moments," *Ind. Mark. Manag.*, vol. 90, no. November 2018, pp. 481–492, 2020, doi: 10.1016/j.indmarman.2019.05.001.
[37]Z. Gharibshah, X. Zhu, A. Hainline, and M. Conway, "Deep Learning for User Interest and Response Prediction in Online Display Advertising," *Data Sci. Eng.*, vol. 5, no. 1, pp. 12–26, 2020, doi: 10.1007/s41019-019-00115-y.
[38]A. A. T. Fernandes, D. B. F. Filho, E. C. da Rocha, and W. da Silva Nascimento, "Read this paper if you want to learn logistic regression," *Rev. Sociol. e Polit.*, vol. 28, no. 74, pp. 1/1-19/19, 2020, doi: 10.1590/1678-987320287406EN.
[39]E. Pekel, "Estimation of soil moisture using decision tree regression," *Theor. Appl. Climatol.*, vol. 139, no. 3–4, pp. 1111–1119, 2020, doi: 10.1007/s00704-019-03048-8.
[40]D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, p. 6, **2020**, doi: 10.1186/s12864-019-6413-7.
[41]S. Albahli, "Efficient hyperparameter tuning for predicting student performance with Bayesian optimization," *Multimed. Tools Appl.*, vol. 83, no. 17, pp. 52711–52735, 2024, doi: 10.1007/s11042-023-17525-w.
[42]T. S. Biró and Z. Néda, "Gintropy: Gini index based generalization of entropy," *Entropy*, vol. **22**, no. 8, pp. **1–13, 2020**, doi: 10.3390/E22080879.
[43]S. Ghosal, S. Sengupta, M. Majumder, and B. Sinha, "Linear Regression Analysis to predict the number of deaths in India due to SARS-CoV-2 at 6 weeks from day 0 (100 cases - March 14th 2020)," *Diabetes Metab. Syndr. Clin. Res. Rev.*, Vol.**14**, No.**4**, pp.**311–315, 2020**, doi: https://doi.org/10.1016/j.dsx.2020.03.017.
[44]M. Saarela and S. Jauhiainen, "Comparison of feature importance measures as explanations for classification models," *SN Appl. Sci.*, Vol.**3**, No.2, p.**272, 2021**, doi: 10.1007/s42452-021-04148-9.
[45]S. Nusinovici *et al.*, "Logistic regression was as good as machine learning for predicting major chronic diseases," *J. Clin. Epidemiol.*, Vol.**122**, pp.**56–69, 2020**, doi: https://doi.org/10.1016/j.jclinepi.2020.03.002.
[46]L. Zhao, S. Lee, and S. P. Jeong, "Decision tree application to classification problems with boosting algorithm," *Electron.*, Vol.**10**, No. 16, pp.**1–13, 2021**, doi: 10.3390/electronics10161903.

AUTHORS PROFILE

Divyanshu Prajapati is currently pursuing a Bachelor of Technology (B.Tech.) degree in Computer Science and Engineering from Jaypee University of Engineering and Technology, located in Guna, Madhya Pradesh, India. His interests include Data mining, Python Programming, and Data science.

Indranil Samantha is currently pursuing a Bachelor of Technology (B.Tech.) degree in Computer Science and Engineering from Jaypee University of Engineering and Technology, located in Guna, Madhya Pradesh, India. His academic and professional interests lie in the fields of Data Analysis, Machine Learning, and Data Analytics, where he is keen to explore innovative solutions and cutting-edge advancements.

Dr. Amit Kumar has completed his Master of Technology in Computer Science and Engineering from Kurukshetra University, Kurukshetra, Haryana, India in year 2002. He has completed his Ph.D. in year 2015 from Jaypee University of Engineering and Technology, Guna in the department of Computer Science and Engineering. The topic of his Ph.D. thesis is "Design and Implementation of an Artificial Immune System for Security of Computers". His areas of interest include Artificial Intelligence, Machine Learning, Soft Computing, Data Science and Computer Immunology. He has published various research papers in international & national journals, presented/published papers in the national conference, seminars, and workshops. He has attended various national/international conferences, workshops, faculty development programs, seminars etc.

Dr. Pankaj Dumka has completed his B. Tech. (Mechanical Engineering) from Inderprastha Engineering College, Ghaziabad in 2007, M. Tech. from Indian Institute of Technology, Kanpur in "Fluids and Thermal Sciences" in 2010, and PhD from Jaypee University of Engineering and Technology, Guna in 2021. He has been working with the Jaypee University of Engineering and Technology as an assistant professor in the Department of Mechanical Engineering since 2011. He has published more than 40 research articles in reputed SCI and SCOPUS indexed Journals. His area of interest includes Thermodynamics, Fluid Dynamics, Computational Fluid Dynamics, Numerical Computations, Python programming, and Solar water desalination.

