Prediction of Advertisement Clicks using Machine Learning and Deep Learning Models

Gadde Ashok
Amrita School of Computing
Amrita Vishwa Vidyapeetham
Coimbatore, India
ashokgadde1350@gmail.com

K Vaisakhkrishnan
Amrita School of Computing
Amrita Vishwa Vidyapeetham
Coimbatore, India
kvaisakhkrishnan@gmail.com

Jai Prakash Reddy D
Amrita School of Computing
Amrita Vishwa Vidyapeetham
Coimbatore, India
reddyjai30@gmail.com

Hema Varshini B
Amrita School of Computing
Amrita Vishwa Vidyapeetham
Coimbatore, India
buddihemavarshini22@gmail.com

Sree Pranavi S
Amrita School of Computing
Amrita Vishwa Vidyapeetham
Coimbatore, India
pranavi.senthilraj@gmail.com

***Abstrac*—Advertising plays a crucial role in the modern economy by helping businesses reach their target customers and promoting their products and services. As the amount of online advertising increases, it becomes increasingly important for businesses to be able to predict which advertisements are most likely to be clicked on by users. In this paper, we investigate the use of machine learning and deep learning models for the prediction of advertisement clicks. We compare the performance of various algorithms on a dataset of advertisement click data, and find that both machine learning and deep learning algorithms are effective for the same. Our results demonstrate the effectiveness of deep learning and machine learning models in predicting advertisement clicks and suggest that they may be a useful tool for businesses seeking to optimize their advertising campaigns.**

***Index Terms*—Advertisement, Prediction, Dataset, Supervised Machine Learning, Unsupervised Machine Learning, Deep Learning, Preprocessing, Regression, Classification**

I. Introduction

Advertising has become an integral part of modern society, with businesses relying on various mediums to promote their products and services to potential customers. One of the most common forms of advertising is online advertising, where businesses place ads on websites and social media platforms in an effort to reach their target audience. With the increasing amount of online advertising, it has become important for businesses to be able to predict which ads are most likely to be clicked on by users. This is because businesses are often charged on a pay-per-click (PPC) basis, where they pay a fee each time an ad is clicked on. Therefore, the ability to accurately predict ad clicks can help businesses optimize their advertising campaigns and improve the return on their investment.

In this paper, we explore the use of Machine learning and Deep learning techniques for the prediction of ad clicks. Machine learning algorithms have been widely used for this purpose, but the advent of Deep learning has led to significant improvements in performance for many tasks. Deep learning models, which are inspired by the structure and function of the human brain, have achieved state-of-the-art results in a variety of fields, including natural language processing, image recognition, and speech recognition. In this study, we compare the performance of machine learning and deep learning models on a dataset of ad click data, and evaluate their effectiveness in predicting ad clicks. Our results provide insights into the capabilities of these approaches for predicting ad clicks and may be useful for businesses seeking to optimize their advertising campaigns.

II. Literature Survey

The paper ”Click-through rate prediction in online advertising: A literature review” [1], is discussing the use of various machine learning models for predicting the click-through rate (CTR) of online advertisements. The authors mention the use of several specific types of models: logistic regression (LR) and polynomial regression (Poly2), as well as factorization machines (FMs), field-aware factorization machines (FFMs), and field-weighted factorization machines (FwFMs).

CTR prediction is an important problem in the field of online advertising, as it helps advertisers understand how likely it is that a user will click on an ad given some context (e.g., the user’s browsing history, the content of the ad, the location of the ad). By predicting CTR accurately, advertisers can optimize the placement and targeting of their ads to maximize the number of clicks and, ultimately, their return on investment.

The models mentioned (LR, Poly2, FMs, FFMs, FwFMs) are all machine learning models that can be trained on data to predict CTR. Logistic regression and polynomial regression are traditional statistical models that make use of linear relationships between variables, while factorization machines are more recent techniques that can model high-dimensional sparse data effectively. FFMs and FwFMs are variations of FMs that take into account the structure of the data (e.g., the relationships between different fields or features) in order to make more accurate predictions.

The paper, ”Neighbour Interaction based Click-Through Rate Prediction via Graph-masked Transformer” [2], mentions three different types of machine learning models: logistic regression (LR), decision trees (DT), and finite state machines (FSM). Logistic regression is a type of statistical model that is often used for binary classification tasks (i.e., predicting a binary outcome such as ”yes” or ”no”). It is based on the idea of finding the best linear relationship between a set of features and the target variable. Decision trees are a type of machine learning model that is often used for classification and regression tasks. They work by dividing the feature space into regions, with each region corresponding to a prediction. Decision trees are called ”trees” because they are made up of a series of interconnected nodes, with each node representing a decision point and each branch representing the outcome of that decision. It looks like you are mentioning three different types of machine learning models: logistic regression (LR), decision trees (DT), and finite state machines (FSM). It’s not clear from your question how these models are related or how they are being used together.

Logistic regression is a type of statistical model that is often used for binary classification tasks (i.e., predicting a binary outcome such as ”yes” or ”no”). It is based on the idea of finding the best linear relationship between a set of features and the target variable.

Decision trees are a type of machine learning model that is often used for classification and regression tasks. They work by dividing the feature space into regions, with each region corresponding to a prediction. Decision trees are called ”trees” because they are made up of a series of interconnected nodes, with each node representing a decision point and each branch representing the outcome of that decision.

Finite state machines (FSMs) are a type of mathematical model used to represent and control the behavior of systems that have a finite number of states. They are often used in the design of computer programs and automated systems to control the flow of execution based on the current state and a set of rules or transitions.

An ensemble is a method for combining the predictions of multiple models in order to make more accurate predictions. A hierarchical ensemble is one in which the models are organized into a hierarchy, with each level of the hierarchy representing a different level of abstraction or granularity. The authors of the paper referenced are proposing a machine learning system that combines these three models in a hierarchical manner to improve the accuracy of CTR prediction.

The paper, ”DHEN: A Deep and Hierarchical Ensemble Network for Large-Scale Click-Through Rate Prediction” [3], talks about the usage of herarchical ensemble stucture for ad click prediction. In a hierarchical ensemble architecture, multiple machine learning models are organized into a hierarchy, with each level of the hierarchy representing a different level of abstraction or granularity. At the top level of the hierarchy, a ”meta-model” makes a final prediction based on the predictions of the models at the lower levels.

The paper, ”Predicting clicks: estimating the click-through rate for new ads” [4], discusses about click-through rate, sponsored search, paid search, Web advertising, CTR, CPC, and ranking.

Click-through rate (CTR) is a measure of the effectiveness of an online advertising campaign for a particular website as a percentage. It is calculated by dividing the number of users who click on a specific link by the number of total users who view a page, email, or advertisement. It is commonly used to measure the success of an online advertising campaign for a particular website as well as the effectiveness of email campaigns.

Sponsored search, also known as pay-per-click (PPC) advertising, is a form of online advertising where advertisers pay a fee each time one of their ads is clicked. It is a way of buying visits to your site, rather than attempting to ”earn” those visits organically.

Paid search is a digital advertising channel that allows businesses to bid for advertising space in the sponsored links section of a search engine’s results page. This is also known as pay-per-click (PPC) advertising, as businesses are charged each time a user clicks on one of their ads.

Web advertising, also known as online advertising, is a form of marketing and advertising which uses the Internet to deliver promotional marketing messages to consumers. It includes email marketing, social media marketing, search engine marketing, and mobile advertising.

Cost-per-click (CPC) is a pricing model used in online advertising, where the advertiser pays a fee each time one of their ads is clicked. It is a way of buying visits to your site, rather than attempting to ”earn” those visits organically.

Ranking is the process of determining the order in which a particular search result appears in the search engine results page. There are various factors that can influence the ranking of a website, including the relevance of the content on the website, the quality of the website’s design, and the popularity of the website.

The paper, ”Practical Lessons from Predicting Clicks on Ads at Facebook” [5] discusses about combining decisions trees and logistic regression for training, and other factors play small roles (though even small improvements are important at scale). Picking the optimal handling for data freshness, learning rate schema and data sampling improve the model slightly, though much less than adding a high-value feature, or picking the right model to begin with.

The paper, ”Ad Click Prediction: a View from the Trenches” [6], is based on FTRL-Proximal online learning algorithm, which has excellent sparsity and convergence properties. The FTRL-Proximal algorithm is an online learning algorithm that can be used for large-scale machine learning problems, particularly those involving sparse data. It is an extension of the Follow-The-Regularized-Leader (FTRL) algorithm, which was introduced by McMahan et al. in 2013.

The FTRL-Proximal algorithm has several nice properties that make it well-suited for use in machine learning. For example, it has strong sparsity and convergence guarantees, which means that it can handle large datasets efficiently and effectively. Additionally, it can adapt to changing data and environments, making it well-suited for use in dynamic settings.

One of the key ideas behind the FTRL-Proximal algorithm is to maintain a separate learning rate for each feature, which allows it to adapt to the sparsity and correlations present in the data. This is achieved by using a ”proximal” term in the update rule, which encourages the learning rates to be small for features with little or no data, and larger for features with more data.

Overall, the FTRL-Proximal algorithm is a powerful tool for online learning and has been applied to a wide range of machine learning tasks, including recommendation systems, natural language processing, and image classification.

The paper, ”Improving Ad Click Prediction by Considering Non-displayed Events” [**?**], discusses about framework for counterfactual CTR prediction to overcome difficulties existing in past models.

Counterfactual CTR prediction refers to the task of predicting the click-through rate (CTR) that would have been observed if a different treatment or intervention had been applied, rather than the one that was actually applied. This is useful in situations where one wants to compare the outcomes of different treatments or interventions, or to understand the causal impact of a particular treatment or intervention on the CTR.

There are several difficulties that past models have faced in trying to perform counterfactual CTR prediction. One difficulty is that it can be difficult to accurately estimate the counterfactual CTR, particularly when the data is noisy or there are confounders present. Another difficulty is that past models often rely on strong assumptions about the data-generating process, which may not hold in practice.

To overcome these difficulties, one possible framework for counterfactual CTR prediction is to use causal inference methods, such as propensity score matching or instrumental variables, to estimate the counterfactual CTR. These methods aim to identify and control for confounders that might affect the relationship between the treatment and the outcome, and can be used to estimate the causal effect of a treatment or intervention on the CTR.

Alternatively, one could use machine learning methods, such as causal trees or causal forests, which are specifically designed to estimate causal effects from observational data. These methods aim to identify and control for confounders by learning complex relationships in the data, and can be used to estimate the counterfactual CTR in a more flexible and robust manner.

III. Data Visualization

Data visualization is the process of creating graphical representations of data in order to gain insights and communicate information effectively. It is a crucial step in the data analysis process, as it allows us to see patterns and trends in the data that may not be immediately apparent from raw numbers alone.

There are many different types of data visualization techniques, ranging from simple bar charts and line graphs to more complex visualizations such as scatter plots and heat maps. The choice of visualization technique depends on the nature of the data and the message that the data is intended to convey.

One of the key benefits of data visualization is that it allows us to process and understand large amounts of data more quickly and effectively. By visually representing data, we can more easily identify patterns, trends, and outliers, which can help us make informed decisions and draw meaningful conclusions from the data.

A. Histogram

A histogram is a graphical representation of the distribution of a continuous variable. It is a bar chart that displays the frequency or probability of different values in the data. Histograms are useful for understanding the underlying distribution of the data and identifying patterns and trends. They can also be used to identify outliers, which are data points that are significantly different from the rest of the data.

Histogram representing Age vs Density (Fig. 1), is shown below. It is observed the the highest density is seen for the age of 30, and the lowest is seen for age less than 20, and greater than 50.

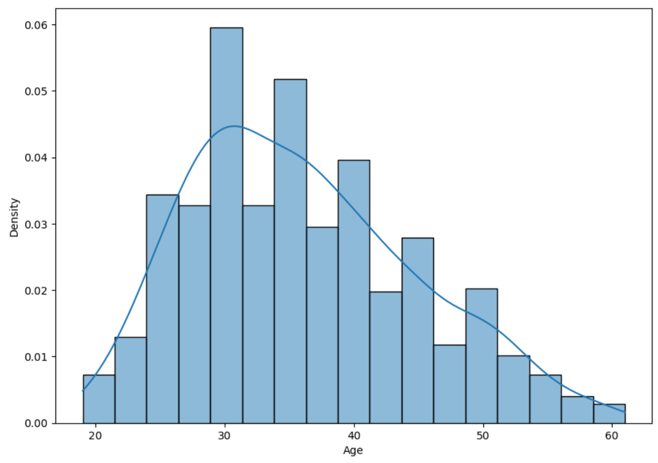


Fig. 1. Histogram of Age vs Density

B. Joint Plot

A joint plot is a graphical representation that combines a scatter plot with additional univariate plots on the top and right sides of the scatter plot, which display the marginal distributions of the variables. It is useful for exploring the relationship between two variables and understanding the underlying distribution of the data. Joint plots can provide insights into the nature of the relationship between the variables, such as whether it is linear or non-linear, and the strength of the association. They can also help to identify any unusual patterns or trends in the data. Overall, joint plots are a useful tool in data science for exploring relationships and understanding the underlying distribution of the data. Joint plot of Age vs Area Income (Fig. 2), and joint plot of Age vs Daily Time Spent on Site (Fig. 3), is shown below

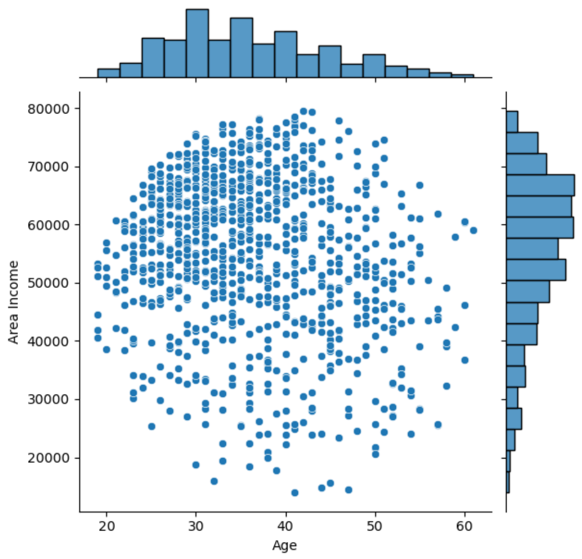


Fig. 2. Joint plot of Age vs Area Income

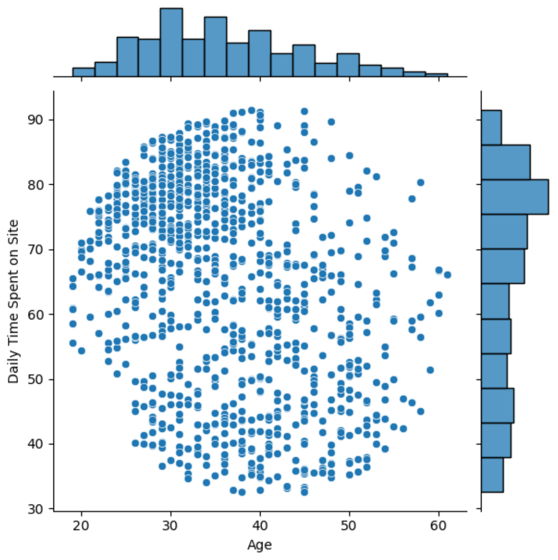


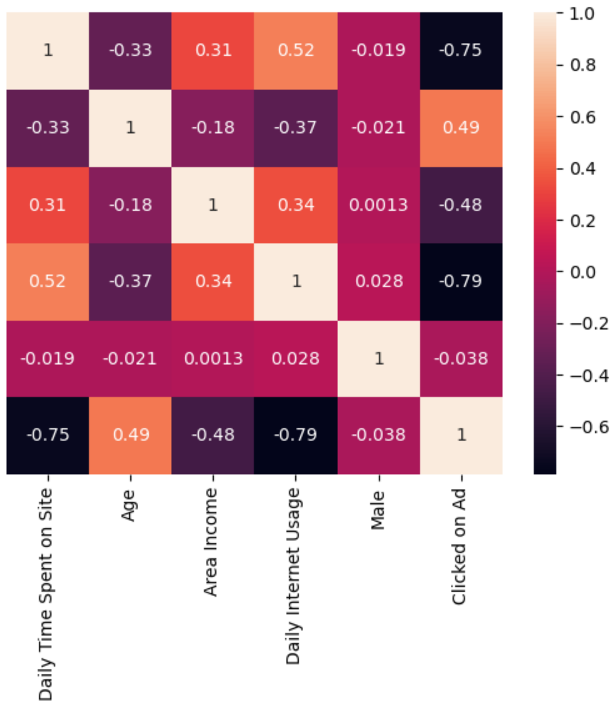
Fig. 3. Joint plot of Age vs Daily Time Spent on Site

C. Heat Map

A heat map is a graphical representation of data where the individual values contained in a matrix are represented as colors. Heat maps are used to visualize patterns and correlations in data, and can be particularly useful for comparing large datasets. Heat map (Fig. 4), representing the correlation of various features with each other is shown below.

D. Box Plot

A box plot, also known as a box and whisker plot, is a graphical representation of a set of numerical data. It displays



	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Male	Clicked on Ad
Daily Time Spent on Site	1	-0.33	0.31	0.52	-0.019	-0.75
Age	-0.33	1	-0.18	-0.37	-0.021	0.49
Area Income	0.31	-0.18	1	0.34	0.0013	-0.48
Daily Internet Usage	0.52	-0.37	0.34	1	0.028	-0.79
Male	-0.019	-0.021	0.0013	0.028	1	-0.038
Clicked on Ad	-0.75	0.49	-0.48	-0.79	-0.038	1

Fig. 4. Heat map representing correlation

the five-number summary of the data: the minimum value, the first quartile, the median, the third quartile, and the maximum value. The box in the plot represents the interquartile range (IQR), which is the range of the middle 50% of the data. The whiskers of the plot represent the minimum and maximum values, excluding any outliers. Outliers are data points that fall outside the range of the IQR, and are plotted separately as individual points. Box plots are useful for quickly visualizing the range, spread, and symmetry of a dataset, and can be helpful for identifying outliers and potential data errors.

Box plot of Area Income (Fig. 5), which represents the minimum, maximum, IQR, and outliers are represented below.

Box plot of Daily Internet Usage (Fig. 6), is represented below.

IV. Feature Engineering

Feature engineering is the process of creating new features or modifying existing features from raw data in order to improve the performance of machine learning models. It is a critical step in the data science process, as the quality of the features used to train a model has a significant impact on its ability to learn and make accurate predictions. There are many different techniques that can be used for feature engineering, including

- Feature selection: choosing a subset of the most relevant features from a larger dataset.
- Feature extraction: creating new features from existing ones using techniques such as principal component analysis (PCA) or singular value decomposition (SVD).

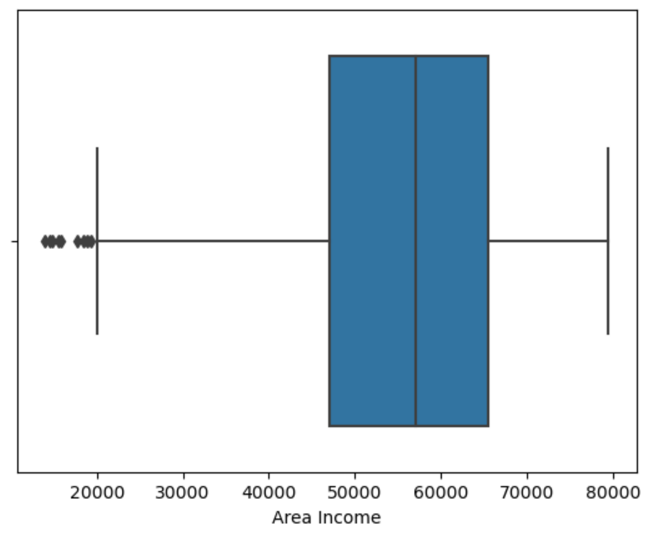


Fig. 5. Box plot of Area Income

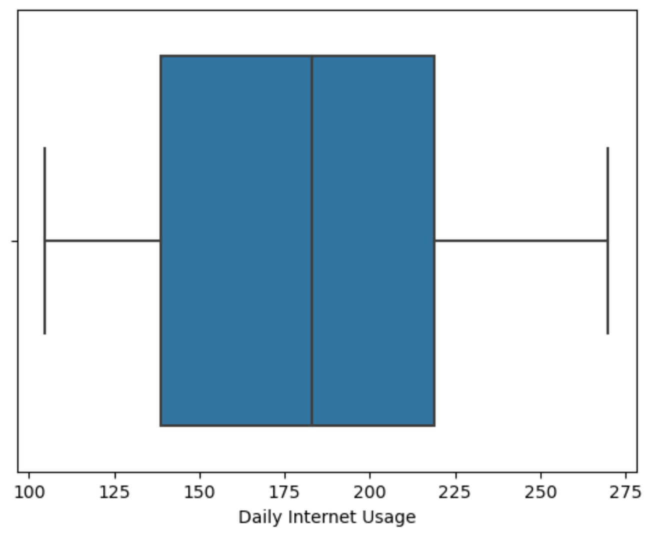


Fig. 6. Box plot of Daily Internet Usage

- Feature transformation: modifying existing features using techniques such as scaling, normalization, or binning.
- Feature creation: creating new features by combining or aggregating existing ones.

A. One Hot Encoding

One hot encoding is a way to represent categorical variables as numerical data. It involves creating a new binary (0 or 1) column for each unique category in a categorical variable. For example, if a categorical variable has three categories (A, B, and C), three new columns would be created. A value of ”A” in the original column would be represented as ”1” in the first new column and ”0” in the other two, a value of ”B” would be represented as ”0” in the first column and ”1” in the second, and so on.

We use One Hot Encoding for City and Country features. Each and every city is converted as a columns and represented in the form of 0’s and 1’s for every row. The same is done in the case of Country.

B. Handling Missing Values

There are many ways to handle missing values. Missing values as=re represented as NaN in the dataset. A NaN value can be handled by either dropping the rows, or by replacing the value with a dummy value.

Here, we prefer replacing the missing values with average or mode of values in a particular column. This is preferred over dropping the values ads dropping rows can result in lesser number of data, as well as might result in dropping relevant data.

C. Feature Scaling

Feature scaling is a preprocessing technique that is used to normalize the range of independent variables or features of a data set. Machine learning algorithms often require that the input features are in the same scale in order to work properly.

For example, consider a dataset with two features, one that ranges from 0 to 100 and another that ranges from 0 to 1. Without feature scaling, the model may be trained with a higher weight on the feature that ranges from 0 to 100, simply because it has a larger range of values. This can lead to poor model performance.

There are two common techniques for feature scaling:

- Min-Max scaling: This scales the data so that the minimum value becomes 0 and the maximum value becomes 1. This is done by subtracting the minimum value from each data point and then dividing the result by the range (max - min).

$$x_1 = \frac{x_0 - min(x)}{max(x) - min(x)} * newrange + newmin$$

- Standardization: This scales the data so that the mean becomes 0 and the standard deviation becomes 1. This is done by subtracting the mean from each data point and then dividing the result by the standard deviation.

$$x_1 = \frac{x_0 - \mu}{\sigma}$$

Here, we are using Min-Max scaling to scale Area Income from the range [55000, 79484] to range [1000, 2000]. This is done because if different features are in different ranges, the model will have less performance.

D. Principal Component Analysis

Principal Component Analysis (PCA) is a statistical technique that is used to reduce the dimensionality of a dataset. It does this by finding a new set of linearly uncorrelated variables, called principal components, which capture the most variance in the data.

PCA works by rotating the original set of variables in order to align the axes with the directions of maximum variance.

The first principal component is the direction that captures the most variance, the second principal component is the direction that captures the next most variance, and so on. The original variables can then be expressed as a linear combination of the principal components.

PCA is often used as a preprocessing step for machine learning algorithms, as it can help to reduce the complexity of the data and improve the performance of the model. It can also be used for data visualization, as it can help to identify patterns and trends in the data. However, PCA can also discard important information in the data, so it is important to consider the trade-offs when using this technique.

To calculate the principal components, the following steps are typically followed:

- Standardize the data: The data should be transformed so that each feature has a mean of 0 and a standard deviation of 1. This is done by subtracting the mean from each value and then dividing the result by the standard deviation.
- Calculate the covariance matrix: The covariance matrix is a square matrix that measures the relationship between the different features. It is calculated by taking the dot product of the standardized data matrix with its transpose.
- Calculate the eigenvectors and eigenvalues: The eigenvectors of the covariance matrix are the directions that capture the most variance in the data. The eigenvalues are the corresponding variances.
- Select the principal components: The number of principal components to select can be determined by selecting a certain number of the highest eigenvalues, or by setting a threshold for the amount of variance that the selected components should capture. The corresponding eigenvectors are the principal components.
- Project the data onto the principal components: The original data can be expressed as a linear combination of the principal components by multiplying the standardized data matrix by the matrix of principal components.

V. Proposed Models

A classifier is a machine learning model that is used to predict a categorical label. It works by learning from a training dataset and making predictions based on that learning. Classifiers are used in a variety of applications, including spam filters, image classification, and medical diagnosis.

There are many different types of classifiers, including:

- Logistic Regression Model
- Naive Bayes Model
- Decision Tree
- K Nearest Neighbors Classifier
- Random Forest
- Multilayer Perceptron
- K Means Clustering
- Agglomerative Clustering

A. *Logistic Regression Model*

Logistic regression is a type of statistical model that is used for binary classification. It is a linear model that is used to predict the probability that a given input belongs to a particular class.

The logistic regression model is based on the logistic function, which is an S-shaped curve that maps any real-valued number to a value between 0 and 1. The logistic function is defined as:

$$p = \frac{1}{1 + e^{-x}}$$

where p is the probability that the input belongs to the positive class, x is the input, and e is the base of the natural logarithm.

Logistic regression is widely used because it is relatively simple and fast to train, and it can be used for binary classification problems with a large number of features. However, it is limited by its assumption that the input features are independent and that the relationship between the features and the labels is linear.

Fig. 7 shows the Sigmoid curve obtained by the Logistic Regression Model

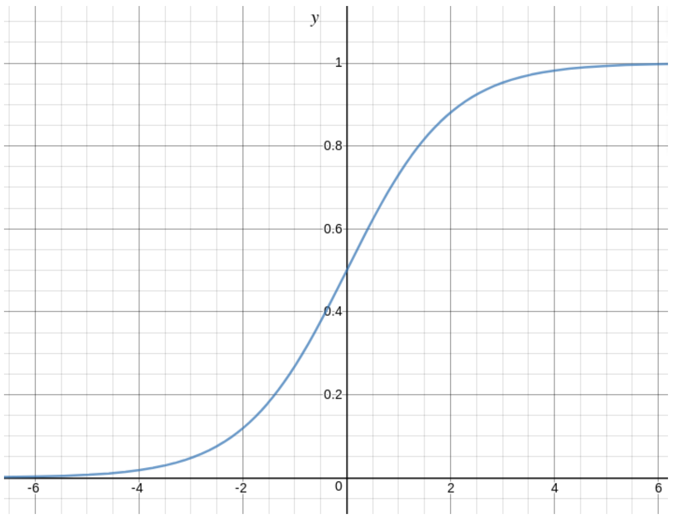


Fig. 7. Sigmoid Curve in Logistic Regression, ref. https://www.geeksforgeeks.org/understanding-logistic-regression

The performance of the classification model is analyzed using Cross Entropy Loss Function. Cross entropy is a loss function that is commonly used in classification tasks. It measures the difference between the predicted probabilities and the true labels, and is defined as:

$$L = -\sum y * log(\hat{y}) + (1 - y) * log(1 - \hat{y})$$

where L is the cross entropy loss, y is the true label (0 or 1), and ŷ is the predicted probability of the positive class (class 1). The sum is taken over all the examples in the dataset.

The cross entropy loss function has the property that it is always non-negative, and it becomes smaller as the predicted probabilities get closer to the true labels. It is often used in conjunction with an optimization algorithm like gradient descent to train a model by minimizing the loss.

One advantage of the cross entropy loss function is that it is easy to compute and differentiate, which is useful for training with gradient-based optimization algorithms. It is also less sensitive to the scale of the predicted probabilities than the mean squared error, which is another commonly used loss function.

The cross entropy loss function is often used for classification tasks because it is a measure of the distance between the predicted probabilities and the true labels, which can be useful for evaluating the performance of a model.

The cross entropy loss function obtained in Logistic Regression is 3.223.

A confusion matrix is a table that is used to evaluate the performance of a classification model. It is used to describe the performance of a classifier on a set of test data for which the true labels are known.

The confusion matrix is a two-by-two table that contains the following four entries:

- True positives (TP): These are the cases where the model predicted the positive class (class 1) and the true label was also class 1.
- True negatives (TN): These are the cases where the model predicted the negative class (class 0) and the true label was also class 0.
- False positives (FP): These are the cases where the model predicted the positive class but the true label was class 0. These are also known as ”Type I errors.”
- False negatives (FN): These are the cases where the model predicted the negative class but the true label was class 1. These are also known as ”Type II errors.”

The confusion matrix is a useful tool for understanding the performance of a classification model and for identifying areas for improvement.

The confusion matrix for logistic regression is shown below (fig. 8)

A Receiver Operating Characteristic (ROC) curve is a graphical plot that is used to evaluate the performance of a binary classifier system as its discrimination threshold is varied. The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) at different classification thresholds.

The area under the ROC curve (AUC) is a measure of the classifier’s overall performance. A classifier with an AUC of 1 has a perfect performance, while a classifier with an AUC of 0.5 has a performance no better than random guessing. ROC curve for Linear Regression model is shown in fig. 9.

B. Naive Bayes Model

Naive Bayes is a type of probabilistic classifier based on the Bayes theorem, which states that the probability of a hypothesis (H) given some evidence (E) is equal to the prior probability of the hypothesis multiplied by the likelihood

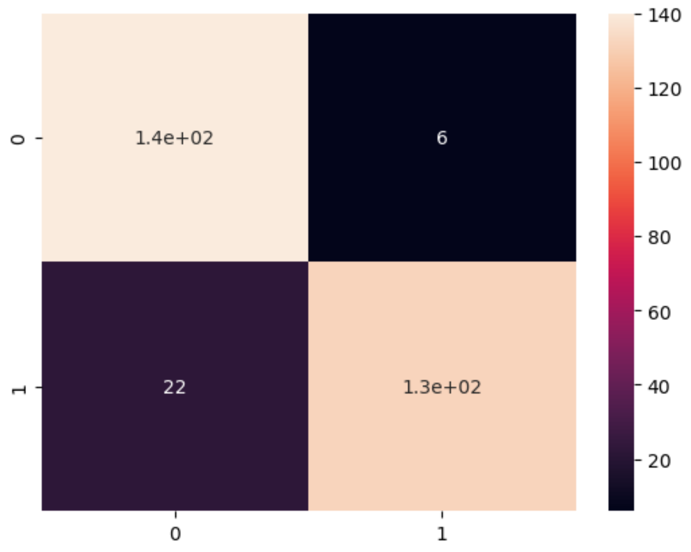


Fig. 8. Confusion Matrix for Logistic Regression

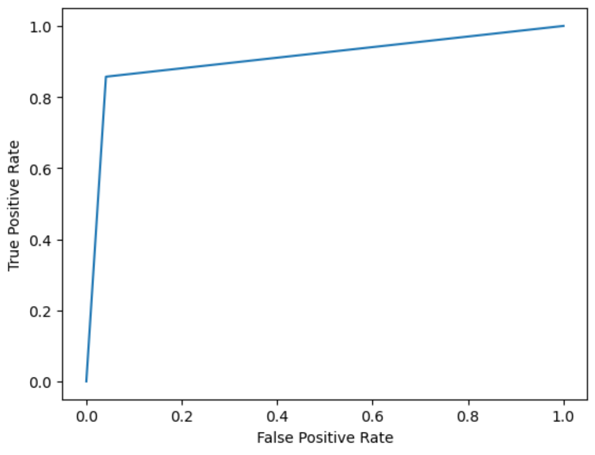


Fig. 9. ROC Curve for Logistic Regression

of the evidence given the hypothesis, divided by the prior probability of the evidence:

$$P(H|E) = (P(E|H) * P(H))/P(E)$$

Naive Bayes classifiers are simple and fast to train, and they can scale to large datasets. They are also resistant to overfitting, which makes them a good choice for high-dimensional data. However, they can be less accurate than other types of classifiers, particularly when the assumption of feature independence is not met.

Fig. 10 shows how Naive Bayes Classifier works.

Our dataset is fitted using Naive Bayes Classifer and we obtained a Cross Entropy Loss Function value of 1.3815.

The Confusion Matrix, and ROC curve for Naive Bayes model are represented in fig. 11 and fig. 12, respectively.

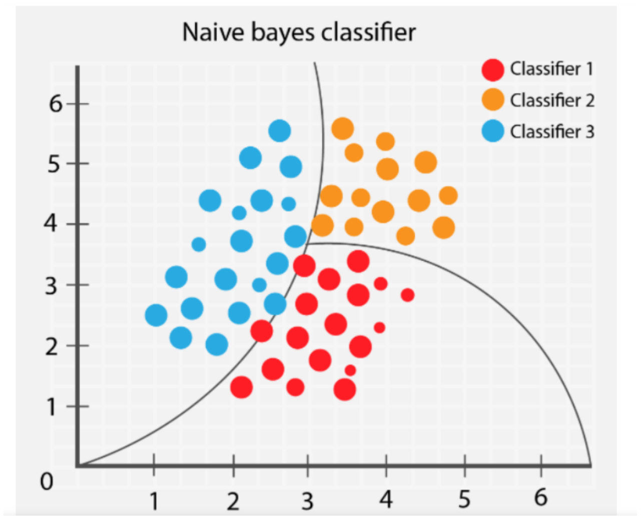


Fig. 10. Sample Naive Bayes Classifier, ref. https://www.analyticsvidhya.com

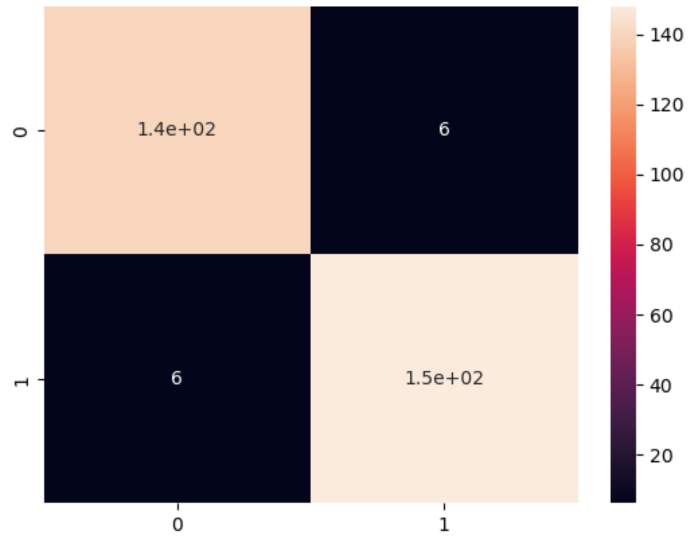


Fig. 11. Confusion Matrix of Naive Bayes model

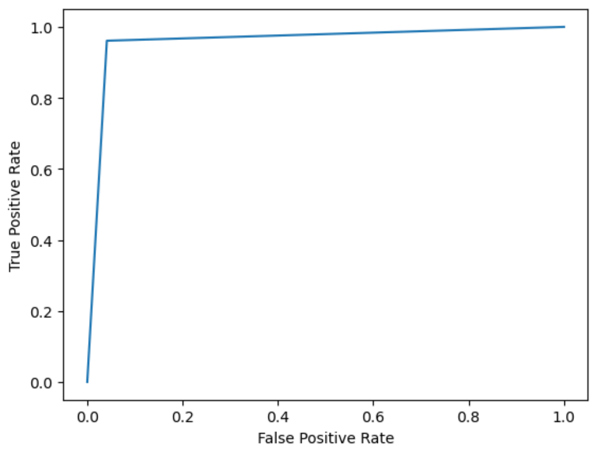


Fig. 12. ROC curve of Naive Bayes model

C. Decision Tree

A decision tree is a flowchart-like tree structure that is used to make decisions based on a series of binary splits. It is a supervised learning algorithm that can be used for classification or regression tasks.

In a decision tree, the root node represents the root of the tree and splits the data into two or more branches based on some decision criteria. The internal nodes represent the decisions that are being made based on the features of the data, and the leaf nodes represent the final decision or prediction made by the tree.

Decision trees are simple to understand and interpret, and they can handle high-dimensional data and missing values. However, they can also be prone to overfitting and may not be as accurate as other types of models.

Fig. 13 shows sample Decision Tree structure.

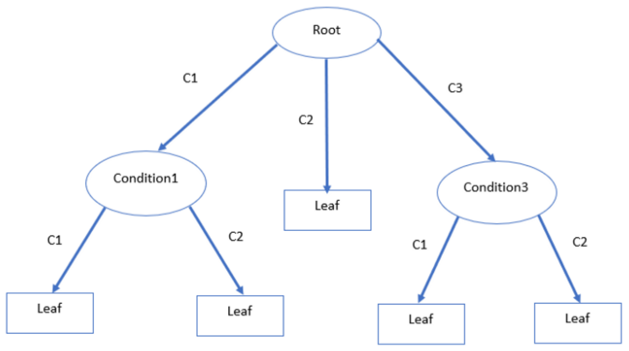


Fig. 13. Sample Decision Tree, ref.: https://regenerativetoday.com

The Confusion Matrix, and ROC curve for Decision Tree model are represented in fig. 14 and fig. 15, respectively.

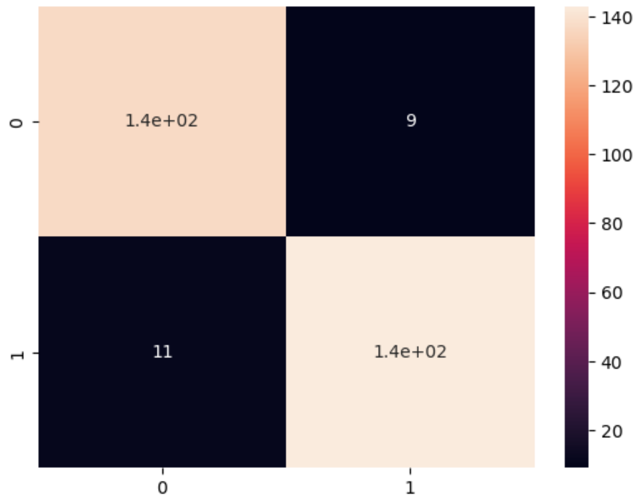


Fig. 14. Confusion Matrix of Decision Tree model

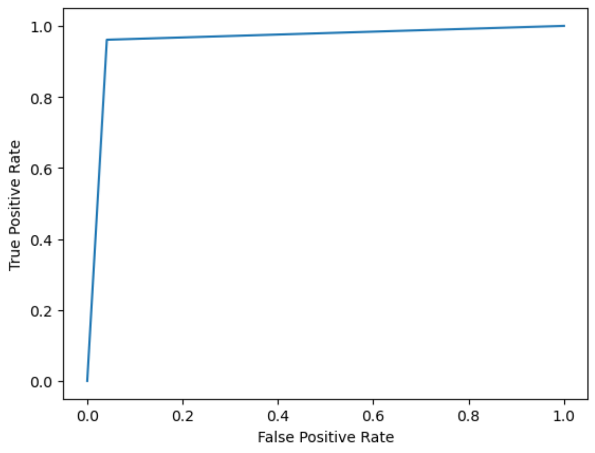


Fig. 15. ROC curve of Decision Tree model

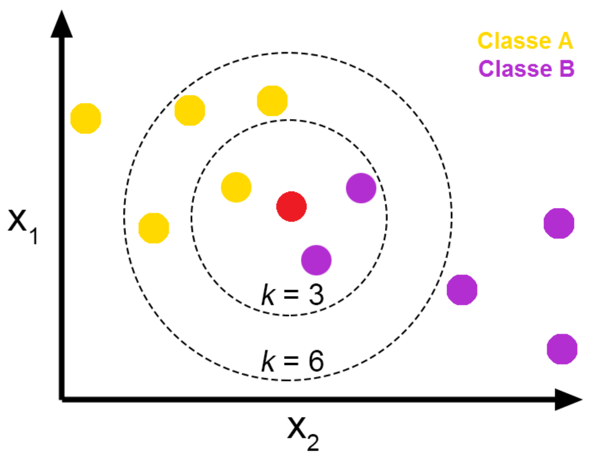


Fig. 16. Sample KNN Model, ref.: https://towardsdatascience.com

D. K Nearest Neighbors Classifier

K-Nearest Neighbors (KNN) is a non-parametric, instance-based, supervised learning algorithm. It is used for classification and regression tasks.

In the KNN algorithm, the training examples are represented as points in an n-dimensional feature space. When a new example is encountered, the algorithm finds the K training examples that are closest to it in the feature space (the ”nearest neighbors”) and predicts the label of the new example based on the labels of those neighbors.

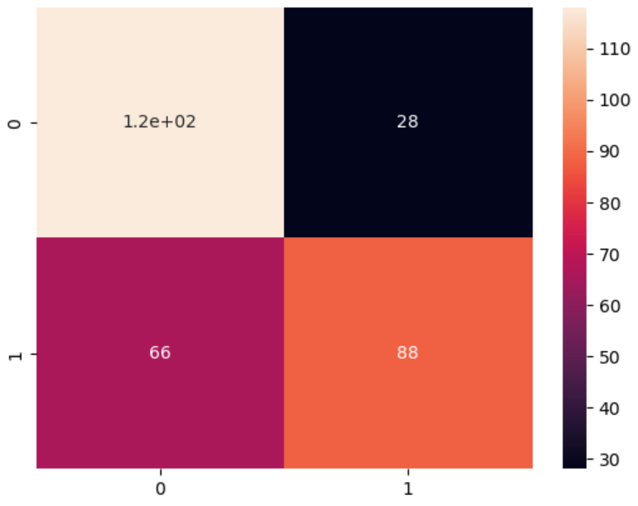
The number of neighbors (K) is a hyperparameter of the model that is chosen by the practitioner. A larger value for K means that the decision boundary will be smoother, while a smaller value will result in a more complex boundary.

One advantage of the KNN algorithm is that it is simple and easy to implement. It also does not make any assumptions about the underlying data, so it can work well with a wide variety of data. However, it can be computationally expensive to find the nearest neighbors, and the model can be sensitive to the choice of K.

Figure 16 shows the working of a basic KNN model

Our dataset was fitted using KNN model and the Cross Entropy Loss value was obtained as 10.822.

The Confusion Matrix, and ROC curve for KNN model are represented in fig. 17 and fig. 18, respectively.



	0	1
0	1.2e+02	28
1	66	88

Fig. 17. Confusion Matrix of KNN model

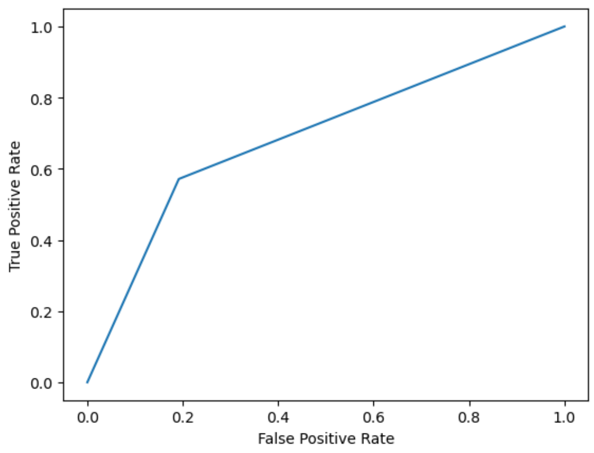


Fig. 18. ROC curve of KNN model

E. Random Forest

A random forest is a machine learning algorithm that is used for classification and regression. It is an ensemble method that uses a collection of decision trees to make predictions. Each tree in the forest is trained on a random subset of the data, and the final prediction is made by averaging the predictions of all the trees. The random forest algorithm is known for its high accuracy, robustness, and versatility. It can handle a large number of features and is resistant to overfitting.

Figure 19 shows the working of a basic Random Forest Model

Our dataset was fitted using Random Forest model and the Cross Entropy Loss value was obtained as 1.95721.

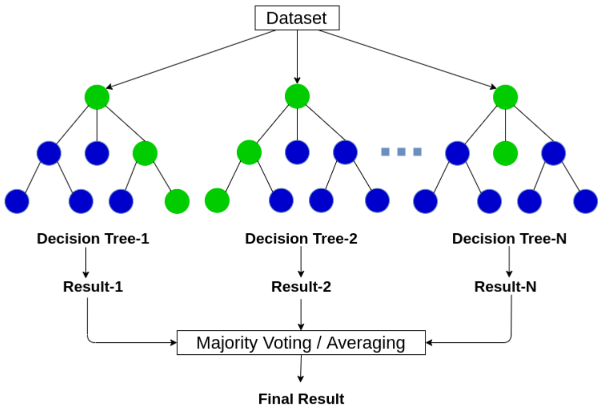


Fig. 19. Sample Random Forest, ref. https://ai-pool.com

The Confusion Matrix, and ROC curve for Random Forest model are represented in fig. 20 and fig. 21, respectively.

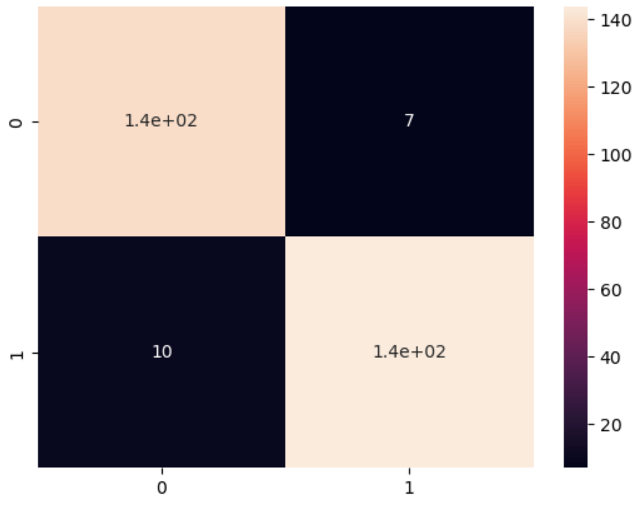


Fig. 20. Confusion Matrix of Random Forest model

F. Multilayer Perceptron

A multilayer perceptron (MLP) is a type of artificial neural network that consists of multiple layers of perceptrons, which are units that compute a linear combination of their inputs and apply a nonlinear activation function to the result. MLPs are feedforward networks, meaning that the information flows through the network in one direction, from the input layer to the output layer, without looping back. MLPs can be used for a wide range of tasks, including classification, regression, and clustering. They are particularly useful for problems that are not linearly separable, meaning that a straight line cannot be drawn to separate the classes. The number of layers and the number of perceptrons in each layer are hyperparameters that can be adjusted to achieve better performance on a specific task.

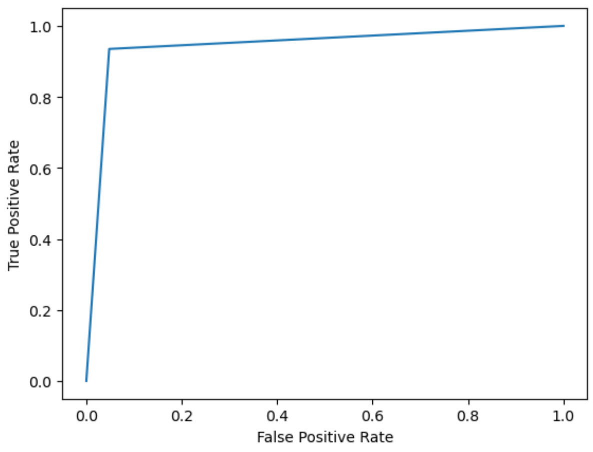


Fig. 21. ROC curve of Random Forest model

Our dataset was fitted into a MLP classifier, and the confusion matrix and Loss curve and ROC curve are represented in fig. 22, fig. 23, and fig. 24, respectively.

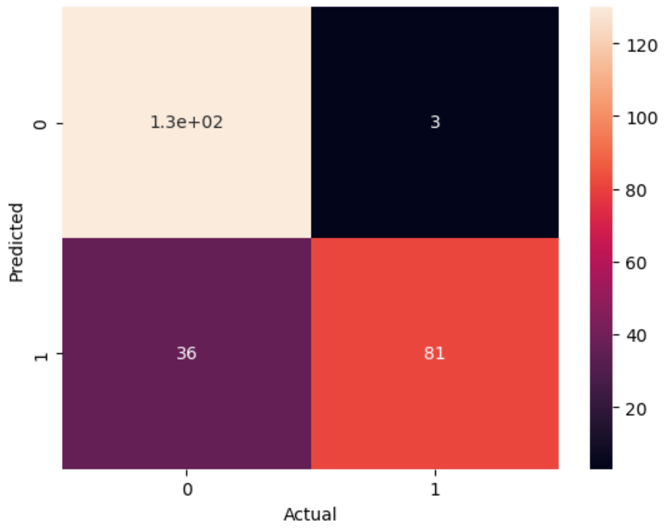


Fig. 22. Confusion Matrix of MLP model

A loss curve is a graph that plots the values of a loss function over time. In the context of training an MLP (multi-layer perceptron), the loss curve can be used to understand how the model is learning and to identify problems such as overfitting or underfitting.

The x-axis of the loss curve represents the number of training iterations or epochs, while the y-axis represents the loss value. As the MLP is trained, the loss should generally decrease, indicating that the model is improving and the predictions are becoming more accurate. However, if the loss is not decreasing or is increasing, it may be a sign of a problem such as overfitting or underfitting.

Overfitting is when the model is too complex and has learned patterns in the training data that do not generalize to new data. This can cause the model to perform well on

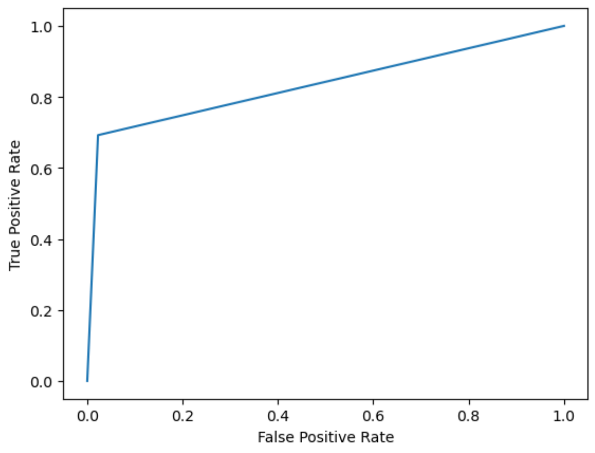


Fig. 23. ROC curve of MLP model

the training data but poorly on test data. In this case, the loss curve for the training data may show a downward trend, but the loss curve for the test data may level off or even increase.

Underfitting is when the model is too simple and cannot learn the necessary patterns in the data. This can cause the model to perform poorly on both the training and test data. In this case, the loss curve for both the training and test data may show a slow downward trend or may not decrease at all.

By examining the loss curve, it is possible to identify problems and adjust the model or training process accordingly to improve its performance.

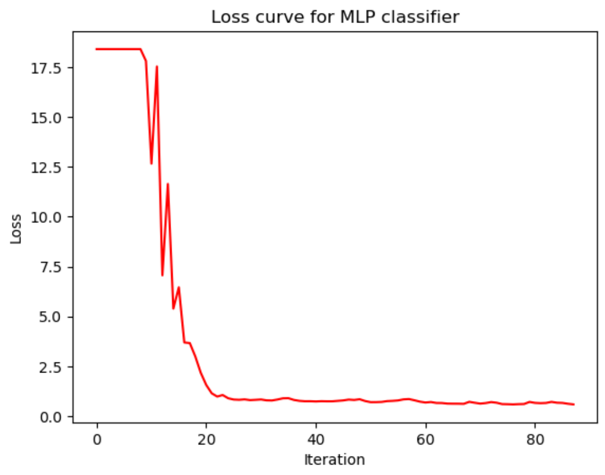


Fig. 24. Loss curve of MLP model

Loss curve for the MLP model is shown in fig. 24. It is seen that as the no of iterations increases, the loss value decreases, and reached very close to 0.

G. K Means Clustering

K-means clustering is an unsupervised machine learning algorithm that is used to divide a set of data points into K clusters, where each cluster is characterized by the mean of the data points that belong to it.

To perform K-means clustering, you first need to specify the number of clusters, K. Then, the algorithm initializes K cluster centroids at random locations. Next, it assigns each data point to the cluster whose centroid is closest to it, based on some distance measure (such as Euclidean distance). After all the data points have been assigned to clusters, the algorithm updates the cluster centroids to the mean of the data points that belong to each cluster. This process is repeated until the centroids no longer move or the improvement in the loss function is below a certain threshold.

One of the main advantages of K-means clustering is that it is computationally efficient and easy to implement. However, it can be sensitive to the initial placement of the centroids and may not always find the true global minimum of the loss function. It is also important to carefully select the value of K, as it can have a significant impact on the resulting clusters.

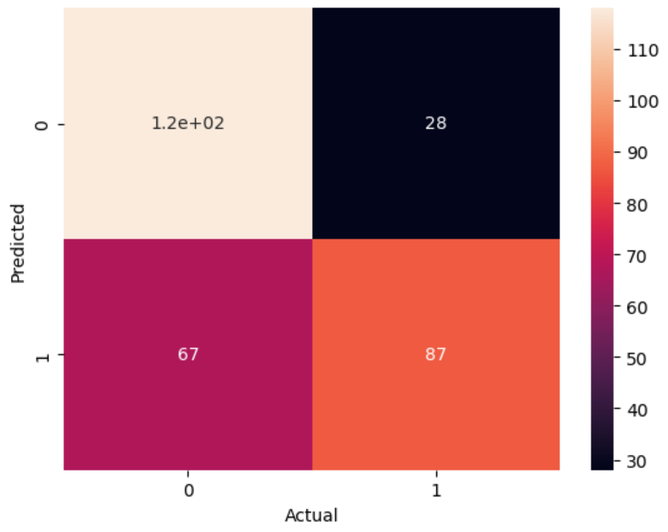


Fig. 25. Confusion Matrix of K Means model

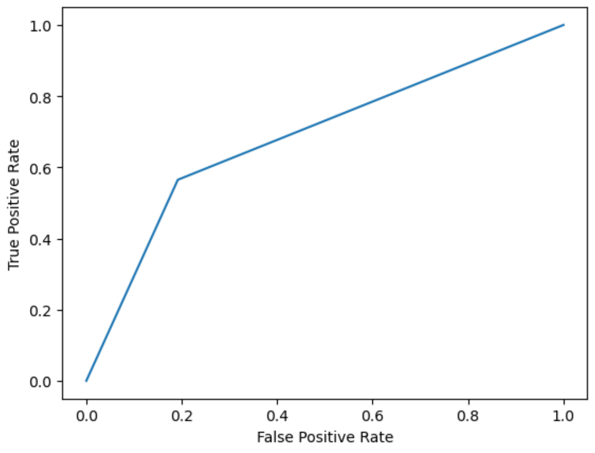


Fig. 26. Loss curve of K Means model

Fig. 25, and fig. 26 shows the confusion matrix and ROC curve for our K Means Classifier.

H. Agglomerative Clustering

Agglomerative clustering is an unsupervised machine learning algorithm that is used to divide a set of data points into clusters. It is a hierarchical clustering algorithm, which means that it builds a hierarchy of clusters, starting with each data point as a single cluster and merging them together as needed.

In agglomerative clustering, the algorithm starts by computing the distance between all pairs of data points. It then combines the two data points that are closest to each other into a single cluster. The distance between this cluster and the other data points is then calculated, and the process is repeated until all the data points have been merged into a single cluster or a predetermined stopping condition has been reached.

One of the main advantages of agglomerative clustering is that it is easy to implement and can handle large datasets. However, it can be computationally expensive, as it requires computing the distance between all pairs of data points. It is also sensitive to the choice of distance measure and can produce clusters that are not as coherent as those produced by other clustering algorithms.

Our dataset was fitted into a Agglomerative classifier, and the confusion matrix and Loss curve and ROC curve are represented in fig. 27, fig. 28, and fig. 24,

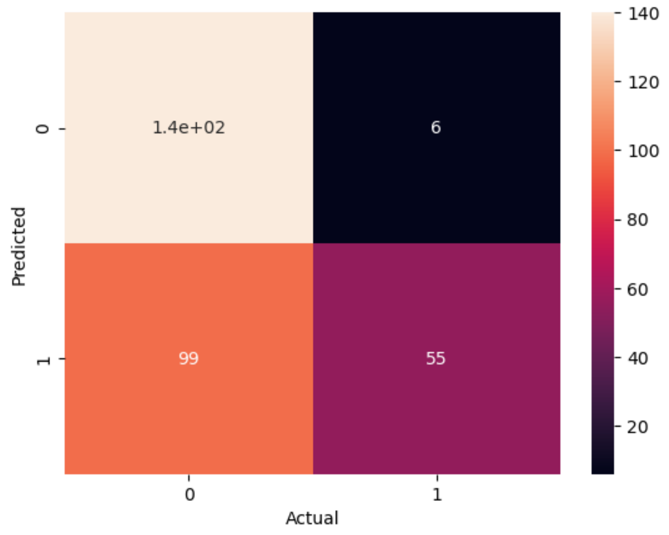


Fig. 27. Confusion Matrix of Agglomerative Clustering

A dendrogram is a graphical representation of a tree-like structure, such as the one produced by hierarchical clustering algorithms. It is used to visualize the hierarchies produced by these algorithms and to understand how the data points are related to each other.

In a dendrogram, each data point is represented by a leaf node and the clusters are represented by the branches that connect them. The length of the branches reflects the distance between the clusters. The dendrogram is read from bottom to top, with the data points at the bottom and the root at the top.

Dendrograms are often used in conjunction with hierarchical clustering algorithms, such as agglomerative clustering, to

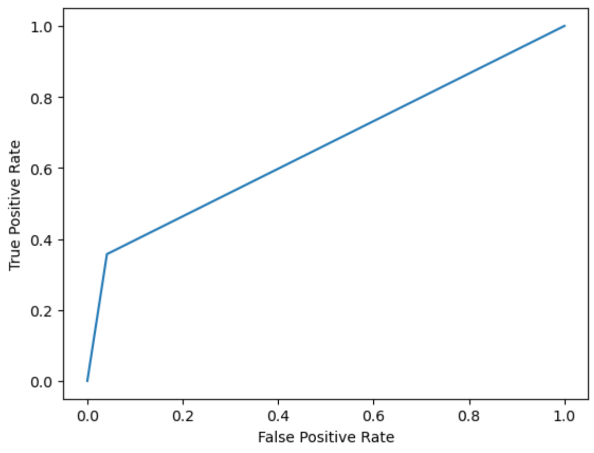


Fig. 28. Loss curve of Agglomerative Clustering

visualize the resulting clusters and to determine the optimal number of clusters. To do this, a horizontal line is drawn through the dendrogram at a level that cuts the tree into the desired number of clusters.

Dendrograms can also be used to visualize other types of hierarchical relationships, such as the evolutionary relationships between organisms or the structure of a computer network.

The Dendogram for our dataset is shown in fig. 29.

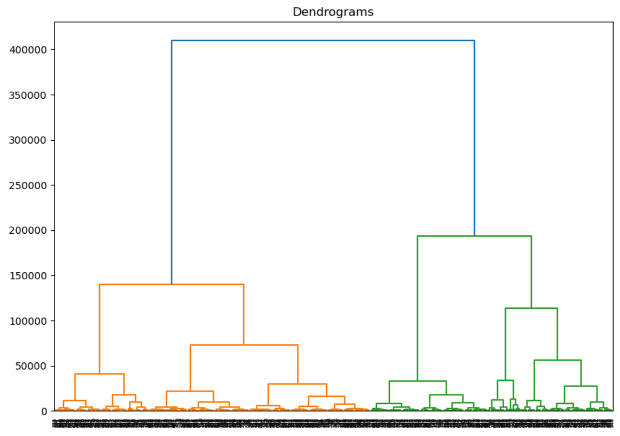


Fig. 29. Dendogram of Agglomerative Clustering

VI. Conclusion

Table I, and table II represents the Precision, Recall, F1-Score, and Performance of various models. Out of various machine learning, and deep learning models tested out, is seen than Naive Bayes and Decision Tree performs much better than any other models. So, it would be better to use either Decision Tree or Naive Bayes model to train the dataset for better performance.

Acknowledgment

We would like to express our gratitude to Dr. S Padmavathi for her invaluable assistance in the development of our Ad

TABLE I
MODEL PERFORMANCE FOR CLASS 0

Model	Precision	Recall	F1-Score	Performance
Logistic Regression	0.86	0.96	0.91	90.66 %
Naive Bayes	0.96	0.96	0.96	96.0 %
Decision Tree	0.93	0.94	0.93	96.0 %
K Nearest Neighbors	0.64	0.81	0.72	68.66 %
Random Forest	0.93	0.95	0.94	94.33 %
Multilayer Perceptron	0.74	1.0	0.70	78.01%
K Means Clustering	0.54	1.0	0.70	68.00 %
Agglomerative Clustering	0.54	1.0	0.68	35.00 %

TABLE II
MODEL PERFORMANCE FOR CLASS 1

Model	Precision	Recall	F1-Score	Performance
Logistic Regression	0.96	0.86	0.90	90.66 %
Naive Bayes	0.96	0.96	0.96	96.0 %
Decision Tree	0.94	0.93	0.93	96.0 %
K Nearest Neighbors	0.76	0.57	0.65	68.66 %
Random Forest	0.95	0.94	0.94	94.33 %
Multilayer Perceptron	1.0	0.41	0.31	78.01%
K Means Clustering	1.0	0.06	0.11	68.00 %
Agglomerative Clustering	1.0	0.06	0.11	68.00 %

Click Prediction Model. Her expertise in Machine Learning and Deep Learning and guidance throughout the process were crucial to the success of this project. We would also like to thank the rest of our team for their hard work and dedication. Finally, we would like to acknowledge the use of the Ad-Click-Prediction dataset in our model. Without it, this work would not have been possible.

REFERENCES

[1] Yanwu Yang, and Panyu Zhai, ”Click-through rate prediction in online advertising: A literature review” Information Processing Management, volume 59, March 2022

[2] Erxue Min, Yu Rong, Tingyang Xu, Yatao Bian, Peilin Zhao, Junzhou Huang, Da Luo, Kangyi Lin, and Sophia Ananiadou, ”Neighbour Interaction based Click-Through Rate Prediction via Graph-masked Transformer” arXiv:2201.13311 [cs.IR], January 2022

[3] Buyun Zhang, Liang Luo, Xi Liu, Jay Li, Zeliang Chen, Weilin Zhang, Xiaohan Wei, Yuchen Hao, Michael Tsang, Wenjun Wang, Yang Liu, Huayu Li, Yasmine Badr, Jongsoo Park, Jiyan Yang, Dheevatsa Mudigere, and Ellie Wen, ”DHEN: A Deep and Hierarchical Ensemble Network for Large-Scale Click-Through Rate Prediction” arXiv:2203.11014 [cs.IR], March 2022

[4] Matthew Richardson, Ewa Dominowska, and Robert Ragno, ”Predicting clicks: estimating the click-through rate for new ads” Association for Computing Machinery, Pages 521–530, May 2007

[5] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, and Joaquin Quiñonero Candela, ”Practical Lessons from Predicting Clicks on Ads at Facebook” Association for Computing Machinery, August 2014

[6] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica, ”Ad click prediction: a view from the trenches” Association for Computing Machinery, August 2013

[7] Bowen Yuan, Jui-Yang Hsia, Meng-Yuan Yang, Hong Zhu, Chih-Yao Chang, Zhenhua Dong, Chih-Jen Lin”Improving Ad Click Prediction by Considering Non-displayed Events” Association for Computing Machinery, 2019