# Assignment 3 Option 2
# Quantity To Produce Quality

A.D. van der Merwe
Department of Computer Science
University of Stellenbosch
24923273
24923273@sun.ac.za

***Abstract*—**

## I. Introduction

## II. Background

This section presents background information on the gradient descent optimisation algorithm, logistic regression model, and ensemble learning. Additionally, background information on bootstrap aggregating, basis functions, and performance metrics used in this report.

### A. Gradient Descent

The gradient descent algorithm was first introduced by Augustin-Louis Cauchy in 1847 [3]. Cauchy introduced gradient descent to solve optimisation systems of simultaneous equations through iterative optimisation to find the minimum of a function. Cauchy also introduced the step size parameter, now commonly referred to as the learning rate, to control how large the steps are for each iteration as the algorithm updates model parameters to reach an optimal solution.

The generic learning algorithm of gradient descent is represented by Algorithm 1.

**Algorithm 1** Gradient Descent Learning Algorithm

1: Preprocess the training set $D_T$ as necessary
2: Initialise parameter vector, $\mathbf{w}(t)$, $t = 0$
3: Initialise the learning rate $\eta$
4: **while** stopping condition not satisfied **do**
5:  **for** each $i = 1, ..., n_T$ **do**
6:  Calculate error signal, $\delta(t)$
7:  Calculate a search direction, $\mathbf{q}(t) = f\left(\mathbf{w}(t), \delta(t)\right)$
8:  Update parameter vector: $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta\mathbf{q}(t)$
9:  **end for**
10:  $t = t + 1$
11:  Compute prediction error
12: **end while**
13: Return $\mathbf{w}(t-1)$ as solution

### B. Logistic Regression

The logistic regression model was first introduced by David Cox in 1958 as a method to perform binary classification [4]. Cox specifically designed the logistic regression model to model the probability of a binary outcome as a function of descriptive features.

To construct a logistic regression model that makes use of gradient descent as an optimisation algorithm, a threshold function that is continuous, and therefore differentiable is needed. This function is known as the logistic function and is represented by the mathematical equation below.

$$Logistic(z) = \frac{1}{1 + e^{-z}} \tag{1}$$

where $z$ is a numeric value.

Before the logistic regression model is constructed the binary target features are mapped to 0 or 1. The logistic regression model is then constructed by use of the equation that follows.

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}_i) = \frac{1}{1 + e^{-\mathbf{w}\cdot\mathbf{d}_i}} \tag{2}$$

where $\mathbf{d}_i$ is a vector of the $i$-th descriptive features, with the bias term represented by $\mathbf{d}_0$ and equal to one, $\mathbf{w}$ is a vector of weights, where $\mathbf{w}_0$ represents the weight of the bias term, and the weights that remain corresponds to their respective descriptive features in $\mathbf{d}_i$. The term $\mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)$ represents the predicted output for the $i$-th instance of the logistic regression model. The output of the logistic regression model can be interpreted as probabilities of the occurrence of a target instance that belongs to a specific class. The probability the $i$-th target instance that belongs to class one is given by the equation below.

$$P(y_i = 1|\mathbf{d}_i) = \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i) \tag{3}$$

where $y_i$ is the true label for the $i$-th observation. Similarly, the probability of the $i$-th target instance that belongs to class zero is given by the equation below.

$$P(y_i = 0|\mathbf{d}_i) = 1 - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i) \tag{4}$$

To classify the $i$-th target instance, $\mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)$ is compared to a threshold of $0.5$. The equation used to classify the $i$-th target feature that belongs to either class zero or class one is given as follows.

$$\hat{y}_i = \begin{cases} 0 & \text{if } \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i) < 0.5 \\ 1 & \text{if } \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i) \geq 0.5 \end{cases} \tag{5}$$

where $\hat{y}_i$ is the predicted class of the $i$-th binary target variable.

Gradient descent is used as the optimisation algorithm to find the optimal decision boundary for a logistic regression model. The optimal decision boundary is defined as the set of weights that minimise the sum of squared error (SSE) based on the training set. The mathematical represntation of the SSE is as follows.

$$L_2\left(\mathbb{M}_{\mathbf{w}}, \mathcal{D}\right) = \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)\right)^2 \tag{6}$$

where $\mathcal{D}$ is the training dataset and $n$ is the number of instances in the training dataset and $L_2$ is the SSE of the training dataset.

The equation used to represent the error signal used in the gradient descent optimisation algorithm to update the weights of the logistic regression model is as follows.

$$\delta(\mathcal{D}, w_j) = \sum_{i=1}^{n}\left((y_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i))\mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)(1 - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i))d_{i,j}\right) \tag{7}$$

where $w_j$ is the $j$-th weight of the logistic regression model and $d_{i,j}$ is the value of the $j$-th feature for the $i$-th instance of the training dataset.

The equation used to update the weights of the logistic regression model by use of the gradient descent optimisation algorithm is as follows.

$$w_j = w_j + \eta\sum_{i=1}^{n}\left((y_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i))\mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)(1 - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i))d_{i,j}\right) \tag{8}$$

where $\eta$ is the learning rate.

The logistic regression model is quite robust to noise and outliers in the dataset. However, the logistic regression can not handle missing values and sensitive to imbalanced classes. Additionally, the logistic regression model requires categorical features to be encoded into numerical representation by either the ordinal encoded or the one hot encoded technique and the data needs to be scaled or normalised. Feature subset selection also applies, as logistic regression tends to overfit on data with a large number of features. The logistic regression also assumes a linear relationship between the descriptive features, where the actual relationship between descriptive features might be non-linear.

### C. Basis Functions

Basis functions are non-linear elements which transforms the linear inputs to the logistic regression into non-linear represntations, while the model itself remains linear in terms of the weights [5]. The addition of basis functions allows logistic regression model to capture relationships between descriptive features which are non-linear.

The data is transformed by use of a series of basis functions, which enables the logistic regression model to effectively manage non-linear relationships between descriptive features. A logistic regression model that makes use of basis functions is represented by the equation below.

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}_i) = \frac{1}{1 + e^{-\sum_{j=0}^{b} w_j \phi_j(\mathbf{d}_i)}} \tag{9}$$

where $\phi_0$ to $\phi_b$ are a series of $b$ basis functions that each transform the $i$-th input vector $\mathbf{d}_i$ in a different way. Usually $b$ is larger than $n$, which means that there are more basis functions than there are descriptive features.

There are several disadvantages when basis functions are used in a logistic regression model to capture non-linear relationships between descriptive features. Firstly, some prior knowledge of these non-linear relationships is required to select appropriate basis functions. Secondly, an increased number of basis functions results in larger gradient descent search spaces, which can lead to longer convergence times and complicate the optimisation process.

### D. Ensemble Learning

Ensemble learning combines several individual models to to obtain better generalisation performance and predict a new instance based on multiple models opposed to a single model [7].

Each model in an ensemble is trained on the same dataset and yields slightly different results due to variations in training data, model configurations or model architectures. Each model performs differently on certain data patterns. By aggregating these diverse models, the ensemble will balance individual errors and achieve better overall performance than any single model alone. This diversity helps the ensemble to cover the limitations of each model, which results in more accurate and robust predictions. An ensemble also helps to decrease the variance in the model predictions.

Approaches used to create these diverse models are

- Train the the same type of model on different subsets of the observation of the training data.
- Train the same type of model which uses different features of the training data.
- Use different types of models, that results in heterogeneous ensembles and cancels out the inductive bias of each model.
- Use different training or optimisation algorithms.
- Use different control parameters
- Use different model architectures.

An ensemble that contains only one type of model is called a homogeneous ensemble.

Ensemble methods usually use one of the voting schemes that follow to aggregate individual predictions [9].

- Majority voting: This method counts the predictions from each model in the ensemble and selects the class with the most votes as the final output.
- Weighted voting: This variant of majority voting assigns a weight to the vote of each model in the ensemble based on the performance or importance of the model. Models with a higher weight has more influence on the final prediction.
- Soft voting: This method is preferred when the output of a model produces probabilities over all classes. The ensemble averages these probabilities to make a final prediction.

### E. Bootstrap Aggregating

Bootstrap aggregating, also know as bagging, was first introduced by Leo Breiman, in 1996, as a method used to generate a diverse set of models to produce an aggregated model [2]. This diverse set of models is formed by training different models on a subset of the original data.

Datasets that differ from one another are created by use of the bootstrapping technique, where the original dataset is sampled with replacement. The datasets that results from this are the same size as the original dataset but contains different instances, where duplicates of instances may be present in the same dataset. This variation enables each dataset to introduce unique patterns, which helps create a diverse set of models in the ensemble.

A validation set can be created from the dataset with the observations that was not included in the bootstrap sample. The error obtained when this validation set is used to prevent the models from overfitting to the training data is known as the out-of-bag (OOB) error estimation.

The equation of the probability that a specific instance is included in a bootstrap sample is as follows

$$\begin{aligned} P(x \in \mathcal{D}_m) &= 1 - \left(1 - \frac{1}{n}\right)^2 \\ P(x \in \mathcal{D}_m) &\approx 1 - e^{-1} \\ P(x \in \mathcal{D}_m) &\approx 0.6322 \end{aligned} \tag{10}$$

where $\mathcal{D}_m$ is the training dataset used by the $m$-th model in the ensemble. On average less than $\frac{2}{3}$ of the instances will appear in a bootstrap sample. This phenomenon could lead to problems when the original dataset is small, as some instances may never be seen in the ensemble. To address the issue of instances that may never be seen in the ensemble, either the size of the bag or the number of models in the ensemble needs to be increased. However, both approaches lead to an increase in the computational complexity of the ensemble.

### F. Performance Metrics

Performance metrics are essential tools when the effectiveness of classification models are evaluated. Performance metrics provide a quantitative measure of how reliable and accurate a prediction model performs classification on a dataset. Key metrics include accuracy, precision, recall, and F1-score, each offering unique insights into different aspects of model performance [1].

*a) Accuracy:* Accuracy is a common method used to evaluate the performance of classification models. The accuracy of a predictive classification model is determined by the proportion of correctly predicted labels against the total number of predictions. The calculation of the accuracy of a predictive model is as follows:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{10}$$

Accuracy is a popular choice of performance measure mainly beacause it is fairly easy to understand and compute. Accuracy generally perform well on well balanced datasets. On imbalanced datasets, accuracy can produce values that are misleading.

*b) Precision and Recall:* Precision is the proportion of true positive (TP) predictions against all of the TP and false positive (FP). The equation to calculate the precision of a classification model is as follows:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{11}$$

Recall is the proportion of TP predictions against all of the TP and false negative (FN). The equation to calculate the recall of a classification model is as follows:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{12}$$

*c) F1-score:* When a binary classification dataset has imbalanced classes, the accuracy of a model can present a high score that does not represent good performance, as the majority group could be overclassified. Therefore, when an imbalanced binary classification dataset is used, it is better to use multiple performance metrics.

The binary F1-score, also known as the Dice similarity coefficient, is the harmonic mean of precision and recall, that provides a balance between the precision and recall [6]. The equation used to calculate the binary F1-score is as follows.

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{13}$$

The binary F1-score proves especially useful when model performance is assessed on imbalanced binary classification datasets.

## III. Implementation

This section provides the approach taken to implement both the linear logistic regression ensemble model and the non-linear logistic regression ensemble model.

### A. Linear Logistic Regression Ensemble

The linear logistic regression model is implemented as described in Section II-B. The logistic regression is then used to create a homogeneous ensemble of linear logistic regression models to enhance predictive accuracy and robustness and the ensemble uses a soft voting scheme to make predictions.

Each model in the ensemble is trained on different subsets of the dataset generated through bootstrapping. Additionally, a random set of features is selected from the original dataset for each model, which further enhances diversity within the ensemble and mitigates the possibility of the linear logistic regression models overfitting to the training dataset as the models are all constructed on diverse feature subsets.

This combination of bootstrapping and feature randomness allows the ensemble to capture a wider range of patterns, which improves the overall predictive performance of the ensemble on imbalanced datasets, as individual predictions aggregate through a soft voting scheme to produce a final output.

An OOB validation set is also constructed to ensure that the models don't overfit on the training data.

The implementation of the linear logistic regression model is provided in Algorithm 2.

**Algorithm 2** Linear Logistic Regression Ensemble Learning Algorithm

Preprocess the training set $D_T$ as necessary
Set parameters: $num\_models$, $\eta$, $iterations$, $bag\_size$, $num\_features$, $patience$
**for** each model $m \in [1, \ldots, num\_models]$ **do**
  Let $n_B = \lfloor bag\_size \times n \rfloor$
  Sample $D_T$ with replacement to create a bagged dataset $D_B$ with the size of $n_B$
  Create a validation set $D'_B$ from the instances not included in $D_B$ for out-of-bag error estimation with $n'_B$ observations
  Randomly select a subset of features $F_S$ from $D_B$
  Use only the randomly selected features to create the subsets $D_S = D_B[:, F_S]$ and $D'_S = D'_B[:, F_S]$
  Add a bias term of 1 to all observations in $D_S$ and $D'_S$
  Initialise $patience\_counter$ = 0 and weights $\mathbf{w}_m$ for model $m$
  $best\_error = \infty$
  **for** iteration $t = 1$ to $iterations$ **do**
    Compute the error signal: $\boldsymbol{\delta}_m(D_S, \mathbf{w}_m) = \sum_{i=1}^{n_B} ((y_i - \mathbb{M}_{\mathbf{w}_m}(\mathbf{d}_i))\mathbb{M}_{\mathbf{w}_m}(\mathbf{d}_i)(1 - \mathbb{M}_{\mathbf{w}_m}(\mathbf{d}_i))\mathbf{d}_i)$ for $\mathbf{d}_i \in D_S$
    Update weights: $\mathbf{w}_m = \mathbf{w}_m + \eta \cdot \boldsymbol{\delta}_m(D_S, \mathbf{w}_m)$
    Compute out-of-bag error: $L_2(\mathbb{M}_{\mathbf{w}_m}, D'_S) = \sum_{i=1}^{n'_B}(y' - \mathbb{M}_{\mathbf{w}}(\mathbf{d}'_i))^2$ for $\mathbf{d}'_i \in D'_S$
    **if** $error < best\_error$ **then**
      $best\_error = error$
      $\mathbf{w}_m^{best} = \mathbf{w}_m$
      $patience\_counter = 0$
    **else**
      $patience\_counter = patience\_counter + 1$
    **end if**
    **if** $patience\_counter \geq patience$ **then**
      **break**
    **end if**
  **end for**
  Store model $(\mathbf{w}_m^{best}, F_S)$ in the ensemble
**end for**

### *B. Non-Linear Logistic Regression Ensemble*

The non-linear logistic regression ensemble is implemented as described in Section III-A with the addition of basis functions as described in Section II-C. The transformation of the linear dataset to a non-linear dataset is given in Algorithm 3.

**Algorithm 3** Transform The Linear Dataset To A Non-Linear Dataset

**function** GENERATEPOLYNOMIALTERMS($n\_features$, $polynomial\_degree$)
  Initialize empty list of terms $T$
  **for** $i \in [1, \ldots, n\_features]$ **do**
    Add single feature term $[i]$ to $T$
  **end for**
  **for** $degree = 2$ to $polynomial\_degree$ **do**
    **for** $i \in [1, \ldots, n\_features]$ **do**
      Add self-interaction term $[i, \ldots, i]$ of length $degree$ to $T$
    **end for**
    **if** $degree > 1$ **then**
      **for** $i \in [1, \ldots, n\_features]$ **do**
        **for** $j \in [i + 1, \ldots, n\_features]$ **do**
          Add interaction term $[i, \ldots, i, j]$ of length $degree$ to $T$
          **if** $degree > 2$ **then**
            Add reverse interaction term $[j, \ldots, j, i]$ of length $degree$ to $T$
          **end if**
        **end for**
      **end for**
    **end if**
  **end for**
  **return** $T$
**end function**

A random subset of the generated basis functions is selected to mitigate the challenge described in Section II-C. This approach addresses the need for prior knowledge of the non-linear relationships in the dataset, which would otherwise be required to select appropriate basis functions. Each logistic regression model is then trained on different subset of features, which is first transformed to be non-linear by use of the $polynomial\_degree$ parameter. A random subset of this newly created non-linear feature set is then selected to ensure to ensure greater model diversity and to capture a broader range of patterns within the data, which enhances the overall predictive performance of the ensemble. The implementation of the non-linear logistic regression model is provided in Algorithm 4.

**Algorithm 4** Non-linear Logistic Regression Ensemble Learning Algorithm

Preprocess the training set $D_T$ as necessary
Set parameters: $num\_models$, $\eta$, $iterations$, $bag\_size$, $num\_features$, $patience$, $num\_nl\_features$, $polynomial\_degree \geq 2$
**for** each model $m \in [1, \ldots, num\_models]$ **do**
    Let $n_B = \lfloor bag\_size \times n \rfloor$
    Sample $D_T$ with replacement to create a bagged dataset $D_B$ with size $n_B$
    Create validation set $D'_B$ from instances not in $D_B$ for out-of-bag error estimation
    Randomly select feature subset $F_S$ from $D_B$
    Use only the randomly selected features to create the subsets $D_S = D_B[:, F_S]$ and $D'_S = D'_B[:, F_S]$
    $terms =$ GENERATEPOLYNOMIALTERMS($|F_S|$, $polynomial\_degree$)
    Transform $D_S$ and $D'_S$ using polynomial terms in $terms$
    $n\_nl = \lfloor num\_nl\_features \times |terms| \rfloor$
    Randomly select $n\_nl$ terms from transformed features
    Update $D_S = D_S[:, n\_nl]$ and $D'_S = D'_S[:, n\_nl]$
    Add bias term of 1 to all observations in $D_S$ and $D'_S$
    Initialize $patience\_counter = 0$ and weights $\mathbf{w}_m$ for model $m$
    $best\_error = \infty$
    **for** iteration $t = 1$ to $iterations$ **do**
        Compute the error signal: $\boldsymbol{\delta}_m(D_S, \mathbf{w}_m) = \sum_{i=1}^{n_B} ((y_i - \mathbb{M}_{\mathbf{w}_m}(\mathbf{d}_i))\mathbb{M}_{\mathbf{w}_m}(\mathbf{d}_i)(1 - \mathbb{M}_{\mathbf{w}_m}(\mathbf{d}_i))\mathbf{d}_i)$ for $\mathbf{d}_i \in D_S$
        Update weights: $\mathbf{w}_m = \mathbf{w}_m + \eta \cdot \boldsymbol{\delta}_m(D_S, \mathbf{w}_m)$
        Compute out-of-bag error: $L_2(\mathbb{M}_{\mathbf{w}_m}, D'_S) = \sum_{i=1}^{n'_B} (y' - \mathbb{M}_{\mathbf{w}}(\mathbf{d}'_i))^2$ for $\mathbf{d}'_i \in D'_S$
        **if** $error < best\_error$ **then**
            $best\_error = error$
            $\mathbf{w}_m^{best} = \mathbf{w}_m$
            $patience\_counter = 0$
        **else**
            $patience\_counter = patience\_counter + 1$
        **end if**
        **if** $patience\_counter \geq patience$ **then**
            **break**
        **end if**
    **end for**
    Store model $(\mathbf{w}_m^{best}, F_S, terms)$ and $n\_nl$ in ensemble
**end for**

## IV. Empirical Procedure

This section of the report outlines the systematic approach used to evaluate the performance of the linear logistic regression ensemble model and non-linear logistic regression ensemble model across multiple binary classification tasks. Additionally, this section provides the approach used to process each dataset to ensure optimal performance of the ensemble models and the performance metrics used to investigate the influence of the number of members in the ensembles, the number of randomly selected features, and the size of the bags. The section provides the control parameters used to construct each linear and non-linear logistic regression ensemble model for each of the binary classification tasks, the experimental setup and the statistical significance tests used to ensure the reliability and statistical soundness of the results.

### A. Performance Metrics

Accuracy is used to investigate the influence of the number of members in the ensembles, the number of randomly selected features, and the size of the bags if the original dataset contains balanced classes. If the original dataset contains imbalanced classes, the F1-score metric is used to investigate the influence of the number of members in the ensembles, the number of randomly selected features, and the size of the bags.

### B. Data Preprocessing

The data has to be preprocessed to ensure optimal and reliable results on each dataset from both the linear logistic regression ensemble model and the non-linear logistic regression ensemble model. The procedures to preprocess each dataset differ. Therefore, each dataset must be explored separately and processed into an optimal form to use in the ensemble models.

*1) Breast Cancer Dataset:* The breast cancer dataset contains numerical features computed from a digitised image of a fine needle aspirate (FNA) of a breast mass, that describe characteristics of the cell nuclei present in the image [8]. The dataset includes a binary target feature labeled 'diagnosis,' which indicates whether the tumor is benign or malignant, denoted by 'B' or 'M', respectively. In total, the dataset consists of 32 features and 569 observations.

The dataset contains a unique identification feature 'id' that is removed from the dataset, as it does not provide meaningful infomation. There are no missing values present in the dataset and the classes are imbalanced.

All descriptive features that remain are numerical and normalised to a range of $[-1, 1]$. The equation used to normalise the descriptive features is as follows

$$d'_{i,j} = \frac{2\,(d_{i,j} - min(\mathbf{d}_j))}{max(\mathbf{d}_j) - min(\mathbf{d}_j)} - 1 \tag{14}$$

where $\mathbf{d}_j$ is a vector of all values of the $j$-th feature, $min(\mathbf{d}_j)$ and $max(\mathbf{d}_j)$ is the minimum and maximum values of the $j$-th feature, respectively and $d'_{i,j}$ is the normalised value of $d_{i,j}$ within the range $[-1, 1]$.

The target feature 'diagnosis' is encoded so that each instance of 'M' is represented as a zero and each instance of 'B' is represented as one. Since the dataset has imbalanced classes, the F1-score serves as the performance metric to evaluate the influence of the number of ensemble members, the number of randomly selected features, and the bag sizes.

*2) Ionosphere Dataset:*

*3) Diabetes Dataset:*

*4) Banana Quality Dataset:*

*5) Spiral Dataset:*

### *C. Experimental Setup*

### *D. Control Parameters*

TABLE I: Linear Logistic Regression Control Parameters

| **Dataset** | **Control Parameters** | | |
|---|---|---|---|
| | ***eta*** | ***epochs*** | ***patience*** |
| ***Breast Cancer*** | 0.00239 | 10678 | 6 |
| ***Ionosphere*** | 0.00012 | 9809 | 6 |
| ***Diabetes*** | 0.01121 | 18690 | 9 |
| ***Banana Quality*** | 0.00023 | 4338 | 5 |
| ***Spiral*** | 0.06708 | 17335 | 7 |

TABLE II: Non-Linear Logistic Regression Control Parameters

| **Dataset** | **Control Parameters** | | | | |
|---|---|---|---|---|---|
| | ***eta*** | ***epochs*** | ***patience*** | ***poly*** | ***% poly*** |
| ***Breast Cancer*** | 0.00013 | 3812 | 5 | 3 | 50 |
| ***Ionosphere*** | 0.0001 | 2797 | 10 | 3 | 70 |
| ***Diabetes*** | 0.00783 | 1000 | 6 | 3 | 80 |
| ***Banana Quality*** | 0.01099 | 5054 | 5 | 3 | 80 |
| ***Spiral*** | 0.00011 | 8755 | 5 | 4 | 60 |

### *E. Statistical Significance and Analysis*

## V. Research Results

## VI. Conclusion

## References


[1] J. Braet, M. Cristina, Hinojosa-Lee, and J. Springael. "Evaluating performance metrics in emotion lexicon distillation: a focus on F1 scores." (2024).

[2] L. Breiman "Bagging predictors." In: Machine learning (1996).

[3] A. Cauchy "Méthode générale pour la résolution des systemes d'équations simultanées." In: Comp. Rend. Sci. Paris (1847).

[4] D. R. Cox "The regression analysis of binary sequences." In: Journal of the Royal Statistical Society Series B: Statistical Methodology (1958).

[5] A. D'arcy, B. Mac Namee, and J. D. Kelleher "Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies." In: MIT press (2020).

[6] B. J. Erickson, and K. Felipe "Magician's corner: 9. Performance metrics for machine learning models." In: Radiology: Artificial Intelligence 3(2021).

[7] M. A. Ganaie, M., Hu, A. K. Malik, M. Tanveer and P. N. Suganthan "Ensemble deep learning: A review." In: Engineering Applications of Artificial Intelligence (2022).

[8] O. Mangasarian, W. Wolberg, N. Street, and W. Street "Breast Cancer Wisconsin (Diagnostic)" In: UCI Machine Learning Repository (1993).

[9] Z. H. Zhou "Ensemble methods: foundations and algorithms." In: CRC press (2012).