**Artificial Intelligence Assignment 1**

**Lesetja Mojapelo**

**30949955**

**Table of Contents**

[**1.** **Introduction** 3](#_Toc144398360)

[**2.** **Literature Review** 3](#_Toc144398361)

[**2.1.** **Gradient Boosting (GBM)** 3](#_Toc144398362)

[**2.2.** **Random Forest** 3](#_Toc144398363)

[**2.3.** **Artificial Neural Networks (ANN)** 3](#_Toc144398364)

[**3.** **Materials and Method** 3](#_Toc144398365)

[**3.1.** **Materials** 3](#_Toc144398366)

[**3.1.1.** **UCI Dataset** 3](#_Toc144398367)

[**3.1.2.** **Variables** 4](#_Toc144398368)

[**3.1.3.** **Records** 4](#_Toc144398369)

[**3.1.4.** **Features** 5](#_Toc144398370)

[**3.1.5.** **Target** 5](#_Toc144398371)

[**3.2.** **Method** 5](#_Toc144398372)

[**3.2.1.** **Preprocessing** 5](#_Toc144398373)

[**3.2.1.1.** **Feature Selection** 5](#_Toc144398374)

[**3.2.1.2.** **Data Scaling** 5](#_Toc144398375)

[**3.2.1.3.** **Label Encoding** 5](#_Toc144398376)

[**3.2.2.** **Evaluation Metrics** 6](#_Toc144398377)

[**3.2.3.** **Train/Validation/Test Split** 6](#_Toc144398378)

[**3.2.3.1.** **Random Forest** 6](#_Toc144398379)

[***3.2.3.2.*** **Gradient Boosting** 6](#_Toc144398380)

[**3.2.3.3.** **Artificial Neural Network** 6](#_Toc144398381)

[**3.2.4.** **Parameters** 6](#_Toc144398382)

[**3.2.4.1.** **Random Forest** 6](#_Toc144398383)

[**3.2.4.2.** **Gradient Boosting** 7](#_Toc144398384)

[**3.2.4.3.** **Artificial Neural Network** 7](#_Toc144398385)

[**4.** **Results and Discussion** 9](#_Toc144398386)

[**4.1.** **Experiment Results** 9](#_Toc144398387)

[**4.1.1.** **Random Forest** 9](#_Toc144398388)

[**4.1.2.** **Gradient Boosting** 10](#_Toc144398389)

[**4.1.3.** **Artificial Neural Network** 10](#_Toc144398390)

[**4.2.** **Best Model** 11](#_Toc144398391)

[**5.** **Conclusion** 11](#_Toc144398392)

[**6.** **References** 12](#_Toc144398393)

# **Introduction**

# **Literature Review**

## **Gradient Boosting (GBM)**

Gradient boosting machines are a supervised tree based machine learning ensemble algorithm that has the primary aim of reducing bias and variance while converting weak learners into strong learners such as to facilitate an improvement in prediction accuracy (Kadiyala & Kumar, 2018). Kadiyala and Kumar (2018) outlines a three-stage process utilised by grading boosting machines, this process involves optimising the loss function, using a weak learner (decision tree), and then applying models that use gradient descent procedures.

## **Random Forest**

Random forest is an ensemble machine learning method, the structure of random forests comprises of a mixture of decision trees whereby the assembly of each tree is embedded in randomness such that a random subset of training data is selected to make decisions based on the features of that random subset (Breiman, 2001). A beneficial characteristic of random forests is their resistance to overfitting, additional insights into the importance of features as praised by Breiman (2001). Random forests are versatile when it comes to handling large datasets though their performance is dependent in how their hyperparameters are tuned which makes a favoured choice because of its balanced blend of structured randomness and adaptability (Breiman, 2001).

## **Artificial Neural Networks (ANN)**

Artificial Neutral Networks (ANNs) are neural networks are computational modelling tools that have been have extensive acceptable in various discipline for their ability to model complex real-world problems. ANNs are structures that are comprised of densely interconnected adaptive simple processing elements (known as artificial neurons) which can perform large parallel computations for the purposes of data processing and knowledge representation (Basheer & Hajmeer, 2000). The goal of ANNs are intended to utilise the characteristics of information processing akin to biological systems such as high parallelism, robustness, fault and failure tolerance, nonlinearity, learning ability, and the capability to handle imprecise and unclear information which is noted by Basheer and Hajmeer (2000) and they go onto praise these characteristics because nonlinearity allows better fitting to data, high parallelism accelerates processing and their ability to learn and adapt allows the system to modify its internal structures as environments change.

# **Materials and Method**

## **Materials**

### **UCI Dataset**

UCI Rice (Cammeo and Osmancik)

(<https://archive.ics.uci.edu/dataset/545/rice+cammeo+and+osmancik>)

### **Variables**

According to Ilkay Cinar and M. Koklu (2019), the data included in the dataset contains morphological features such as:

|  |  |
| --- | --- |
| Name | Description |
| Area | The number of pixels within the boundaries of the rice grain |
| Perimeter | The circumference by calculating the distance between pixels around the boundaries of the rice grain. |
| Major Axis Length | The longest line that can be drawn on the rice grain, i.e., the main axis distance, gives. |
| Minor Axis Length | The shortest line that can be drawn on the rice grain, i.e., the small axis distance, gives. |
| Eccentricity | It measures how round the ellipse, which has the same moments as the rice grain, is |
| Convex Area | Returns the pixel count of the smallest convex shell of the region formed by the rice grain. |
| Extent | Returns the ratio of the region formed by the rice grain to the bounding box. |
| Class | Cammeo and Osmancik. |

### **Records**

#### **Number of instances**

The dataset contains 3810 instances, these instances have eight columns.

#### **Class Distribution**

|  |  |
| --- | --- |
| Class | Count |
| Cammeo | 1630 |
| Osmancik | 2180 |

#### **Missing Values**

The chosen data set has no missing values.

#### **Outliers**

With a threshold of three, the identified outliers in the dataset for columns appears as follows:

A graph with red and blue lines

Description automatically generated

The above graph indicates that the dataset contains 40 outliers.

### **Features**

The first seven columns in the dataset will be designated as features and will appears as follows:

1. Area
2. Perimeter
3. Major Axis Length
4. Minor Axis Length
5. Eccentricity
6. Convex Area
7. Extent

### **Target**

The last column in the dataset is designated as the class and this class is has the following class values:

1. Class
   1. Cammeo
   2. Osmancik

## **Method**

## **Preprocessing**

### **Feature Selection**

All the features in the dataset were included based on the assumption that each of the features are relevant to classifying rice grains between Cammeo and Osmancik. Additionally, no feature engineering was applied.

### **Data Scaling**

The dataset was scaled using standardisation methods so that features have a mean of 0 and standard deviation of 1.

### **Label Encoding**

The ‘Class’ column in the dataset was labelled into a binary format, where Cammeo is encoded to 0 and Osmancik to 1. The code snippet below illustrates this process.



## **Evaluation Metrics**

* Accuracy – Measures the overall effectiveness of the model in classifying Cammeo and Osmancik rice grain classes.
* Precision – Evaluates the quality of the model.
* Recall – Asses the sensitivity of the model
* F1 Score – Used to indicate the balance between Precision and Recall
* ROC Curve and AUC – Receiver Operating Characteristics Curve plots the true positive rate against the false positive rate. Area Under Curve quantifies the overall ability if the model in discriminating between the classes.
* Confusion Matrix - Provides a detailed breakdown of the model’s classification performance by showing true positives, true negatives, false positives, and false negatives.

## **Train/Validation/Test Split**

### **Random Forest**

A train/test split ratio of 70-30 was respectively used for training and testing. No validation split set was used but instead an Out-of-Bag Score was utilised to estimate generalisation errors.

### **Gradient Boosting**

A train/validation/test split ratio of 70-15-15 was respectively used for training, validation, and testing.

### **Artificial Neural Network**

A train/validation/test split ratio of 70-15-15 was respectively used for training, validation, and testing.

## **Parameters**

### **Random Forest**

In three distinct configurations, the model is fine-tuned through various hyperparameters to optimize its performance. Initially, the model employs default settings, except for the Out-of-Bag (OOB) score, which allows for an unbiased calculation of generalization accuracy. Building upon this foundation, the second configuration introduces additional hyperparameters: the criterion is set to 'entropy' to guide the quality of node splits, the maximum number of features considered at each split is set to the logarithm to base 2 of the total number of features ('log2'), and the forest is specified to contain 50 trees ('estimators=50'). The third and final configuration imposes further constraints on the model to mitigate overfitting. It limits the maximum depth of each tree to 10 and mandates a minimum of 10 samples for leaf nodes ('min\_samples\_leaf=10'). These refinements aim to produce a more generalized and simplified model.

#### **Trial 1**



#### **Trial 2**



#### **Trial 3**



### **Gradient Boosting**

The sequence of gradient boosting models is designed to fine-tune the overall model's performance incrementally. The initial trial employs a hundred trees and serves as the foundation. This is followed by the introduction of a conservative learning rate of 0.1%, aimed at enhancing the model's training precision, although it leads to slower convergence. Building on this foundation, the subsequent model increases complexity by extending the number of trees to 250. It also introduces a maximum tree depth of 10 and sets a minimum sample size of 15 for the leaf nodes. To strike a balance between learning speed and accuracy, this more complex model adopts a higher learning rate of 5%.

#### **Trial 1**



#### **Trial 2**



#### **Trial 3**



### **Artificial Neural Network**

#### **Model Architecture**

The designed neural network features an architecture with three hidden layers, containing 128, 64, and 32 neurons, respectively. Each of these layers employs the Rectified Linear Unit (ReLU) activation function for non-linearity. The input layer is comprised of seven neurons, corresponding to the seven features present in the dataset. To mitigate the risk of overfitting, each architecture trial incorporates two dropout layers with a dropout rate set at 40%. The output layer consists of a single neuron and employs a sigmoid activation function, making it well-suited for the task of binary classification. All neurons in the network will be initialized using a normal weight distribution.

#### **Model Configuration**

In alignment with its binary classification objective, the model utilizes a binary cross-entropy loss function. The optimization strategy for each trial focuses exclusively on tuning the optimizer, which operates with a learning rate of 8.5%. The model's performance will be evaluated using accuracy as the key metric.

#### **Trial 1**



#### **Trial 2**



#### **Trial 3**



# **Results and Discussion**

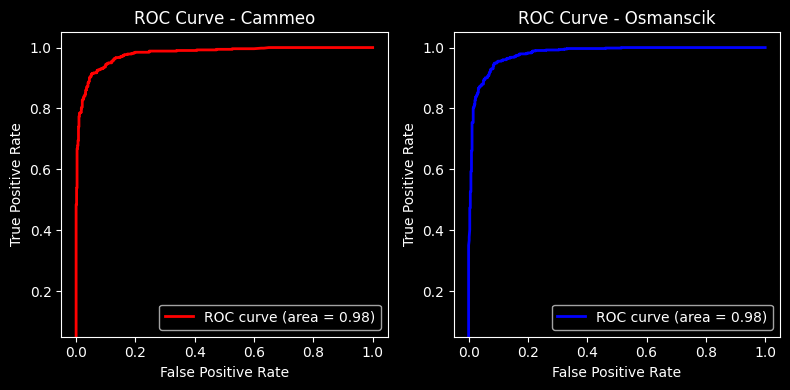
## **Experiment Results**

### **Random Forest**

#### **Classification Report**



#### **ROC Curve and AUC**



#### **Confusion Matrix**

A screenshot of a graph

Description automatically generated

#### **Discussion**



With the above hypermeters from the third random forest trial, the Precision, Recall, F1-Score and Accuracy metrics from the classification report indicate the following for the two classes:’

Class 0 (Cammeo):

* Precision: 94%
* Recall: 91%
* F1-Score: 92%

Class 1 (Osmancik)

* Precision: 92%
* Recall: 95%
* F1-Score: 94%

The accuracy of the model

The ROC Curves near the top left border of the graph which is an indication that the model performs well with the respective classes and does well to distinguish between positive and negative instances. The AUC has an area of 98% which is implies that the model can distinguish between chosen positive and negative instances. Additionally, the model correctly identified 469 instances of Cammeo and 593 instance of Osmancik while it incorrectly classified 49 instances of Cammeo and 32 instances of Osmancik.

### **Gradient Boosting**

#### **Classification Report**



#### **ROC Curve and AUC**

A screenshot of a graph

Description automatically generated

#### **Confusion Matrix**

A screenshot of a computer

Description automatically generated

#### **Discussion**



With the specified hyperparameters in the CatBoostClassifier model, the classification metrics offer the following insights for the two classes

Class 0 (Cammeo):

* Precision: 91%
* Recall: 91%.
* F1-Score: 91%.

Class 1 (Osmancik)

* Precision: 94%
* Recall: 94%
* F1-Score: 94%

The accuracy of the model correctly classifies 93% of all instances, indicating strong overall performance. The ROC Curves near the top left border of the graph which is an indication that the model performs well with the respective classes and does well to distinguish between positive and negative instances. The AUC has an area of 98% which is implies that the model can distinguish between chosen positive and negative instances. Additionally, the model correctly identified 427 instances of Cammeo and 633 instance of Osmancik.

### **Artificial Neural Network**

#### **Classification Report**

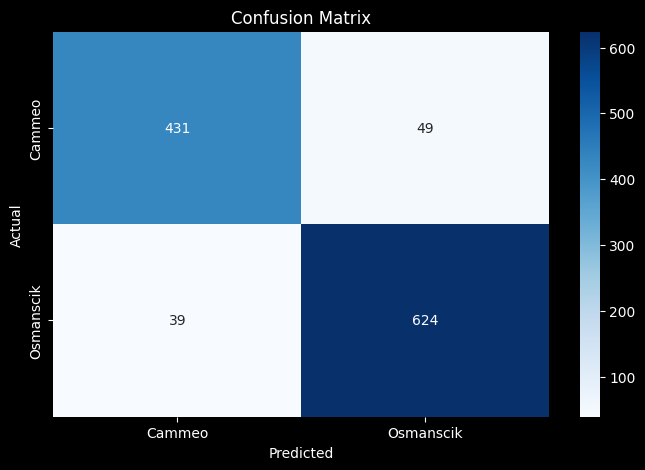


#### **ROC Curve and AUC**

A screenshot of a graph

Description automatically generated

#### **Confusion Matrix**



#### **Discussion**



With the specified hyperparameters in the Artificial Neural Network (ANN) model, the classification metrics yield the following insights for the two classes.

Class 0 (Cammeo)

* Precision: 91%
* Recall: 91%
* F1-Score: 91%.

Class 1 (Osmancik)

* Precision: 94%
* Recall: 94%
* F1-Score: 94%.

Overall, the model has an accuracy 93%. The confusion matrix reveals 431 true positives and 49 false negatives for Class 0, along with 624 true positives and 39 false negatives for Class 1. Furthermore, the ROC Curve is situated near the top-left corner of the graph, and the AUC is 98%. These metrics collectively suggest that the ANN model is capable of distinguishing between the two classes and is likely to perform exceptionally well on unseen data.

## **Best Model**

The best model is the random forest model. The reason for this is because has the highest precision for classifying Cammeo and the second highest recall for also classifying cammeo. It also identified more instances for both classes when compared to the other model.

Another reason for this is because random forest performed well all round when considering that it did well to work with the imbalance in Cammeo vs Osmancik as the other models were better at classifying Osmancik more than Cammeo given that Osmancik has more instances in the dataset which became their weak point. On the other hand, this imbalance did not have any effect on random forest as it is skilled in handing duh imbalances.

# **Conclusion**

All three models—Random Forest, Artificial Neural Network (ANN), and CatBoostClassifier—exhibit exceptional performance in classifying the two given classes. Each model achieves an accuracy of 93% and an AUC of 98%, indicating strong capabilities in both overall classification and distinguishing between positive and negative instances. However, the Random Forest model edges out slightly in terms of precision and recall figures and the number of correctly identified instances, making it the top-ranked model among the three. The ANN and CatBoost models are closely matched in their performance, with nearly identical precision, recall, and F1-scores, and therefore share the second position in this ranking. Given these metrics, any of these models could be a strong candidate for deployment, but the Random Forest model takes a slight lead based on the criteria we've considered.

# **References**

Basheer, I. A. and Hajmeer, M. 2000. Artificial neural networks: Fundamentals, computing, design, and application. *Journal of Microbiological Methods,* 43(1):3-31.

Breiman, L. 2001. Random forests. *Machine Learning,* 45(1):5-32.

Ilkay Cinar and M. Koklu (2019) 'Rice (cammeo and osmancik)' (Accessed: 25 August 2023).

Kadiyala, A. and Kumar, A. 2018. Applications of python to evaluate the performance of decision tree-based boosting algorithms. *Environmental Progress & Sustainable Energy,* 37(2):618-623.