**Members:**

Eman Butt(SP23-BAI-014)

Ayesha Tahir(SP23-BAI-019)

**Submitted to:**

Dr.Aksam Iftikhar

**Course:** Programming for AI

**Project:** Implementation of Algorithms



**Department of Computer Science**

**COMSATS University Islamabad**

**Lahore Campus**

**Minecraft Screenshot Analysis: Classification of Mob Decisions:**

**1. Introduction:**

The dataset used in this project, **"Minecraft\_Screenshots\_Analysis.csv,"** contains images from the game Minecraft, labeled with decisions made by mobs (in-game creatures). The task is to predict these decisions based on extracted features from the images. In the context of this analysis, we aim to apply various machine learning algorithms to classify the mob decisions (label: decision\_mob) based on input features. The project involves using classification models and evaluating their performance through various metrics.

**2. Methodology:**

**2.1 Data Preprocessing:**

The preprocessing steps performed on the dataset are as follows:

* **Data Import and Inspection**: The dataset was imported using pandas and the first few rows were examined.
* **Handling Categorical Features**: Label encoding was applied to all categorical columns in the dataset. This transforms categorical variables into numeric ones for machine learning models.
* **Handling Missing and Duplicate Data**: The dataset was checked for null values and duplicates, and duplicates were removed.
* **Feature Scaling**: The feature set (X) was standardized using StandardScaler to ensure that all features have the same scale.
* **Splitting Data**: The dataset was split into training and testing sets with an 80/20 split using train\_test\_split.

**2.2 Algorithms Applied:**

The following machine learning algorithms were used in this analysis:

1. **Random Forest Classifier**: A Random Forest model was trained and tuned using both GridSearchCV and RandomizedSearchCV to find the best hyperparameters. The model was then evaluated using the accuracy score and AUC-ROC.
2. **Logistic Regression**: A logistic regression model was tuned similarly with GridSearchCV and RandomizedSearchCV, using different hyperparameters to find the best fit.
3. **XGBoost**: XGBoost, a gradient boosting model, was applied using both grid search and randomized search for hyperparameter tuning.

**2.3 Optimization Techniques:**

* **Hyperparameter Tuning**: For each algorithm, hyperparameter optimization was performed using GridSearchCV and RandomizedSearchCV. These methods help identify the best combination of hyperparameters to improve model performance.
* **Cross-validation**: Cross-validation was used in the optimization process to ensure the robustness and reliability of the model evaluation.

**3. Results:**

**3.1 Performance Metrics:**

* **Accuracy**: The accuracy of the models on the test set was reported. **Random Forest** achieved an accuracy of **0.99**, **Logistic Regression** achieved an accuracy of **0.88**, and **XGBoost** achieved an accuracy of **0.99**.
* **Precision, Recall, F1-Score**: These metrics were calculated for each algorithm to evaluate how well each model handled the positive and negative classes. The Random Forest and XGBoost models achieved high scores across these metrics (around 0.99), whereas Logistic Regression had slightly lower values.
* **ROC AUC**: The Area Under the Receiver Operating Characteristic Curve (ROC AUC) was calculated, which reflects the model's ability to distinguish between the classes. Random Forest and XGBoost performed well, achieving an AUC of 0.99, while Logistic Regression achieved an AUC of 0.96.

**3.2 Visualizations:**

* **Confusion Matrix**: A confusion matrix was generated for each model to visually assess the classification performance. The heatmaps for each model provide insight into the model's ability to correctly predict classes.
* **Bar Plot**: A bar chart was created to compare the performance of all three algorithms (Random Forest, Logistic Regression, and XGBoost) based on accuracy, precision, recall, and F1-score.

**4. Analysis**

**4.1 Insights:**

* **Algorithm Performance**: Random Forest and XGBoost performed similarly well, with high accuracy, precision, recall, and F1-scores. Logistic Regression, while still performing decently, lagged slightly behind, likely due to its simpler nature compared to the other two ensemble methods.
* **Hyperparameter Tuning Impact**: The hyperparameter optimization, especially using RandomizedSearchCV, significantly improved model performance by finding the best parameter set for each algorithm.
* **Model Robustness**: Both Random Forest and XGBoost exhibited robust performance, making them suitable for the task. Logistic Regression, while simpler and faster to train, might not capture the complexity of the decision-making patterns as effectively as the ensemble methods.

**4.2 Algorithm Comparison:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Precision** | **Recall** | **F1-Score** | **ROC AUC** | **Best Hyperparameters** | **Execution Time (s)** | **Remarks** |
| **Random Forest** | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | n\_estimators: 20, min\_samples\_split: 10, min\_samples\_leaf: 1, max\_features: log2, max\_depth: None, bootstrap: False | 128 | Achieved excellent overall performance with hyperparameter optimization and high stability. |
| **Logistic Regression** | 0.88 | 0.87 | 0.88 | 0.88 | 0.96 | solver: saga, penalty: elasticnet, max\_iter: 500, l1\_ratio: 0.1, C: 10 | 60 | Delivered decent results, with slightly lower performance compared to Random Forest and XGBoost. |
| **XGBoost** | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | subsample: 0.9, n\_estimators: 100, min\_child\_weight: 2, gamma: 0.1, learning\_rate: 0.1, max\_depth: 5 | 73 | Matched Random Forest in performance, excelling with fast execution time and hyperparameteroptimization. |

**4.3 Challenges:**

* **Class Imbalance**: While the dataset doesn't show severe class imbalance, models like Logistic Regression may still suffer when handling classes with fewer samples. More advanced techniques such as oversampling or undersampling could improve its performance.
* **Tuning Time**: Hyperparameter tuning with GridSearchCV and RandomizedSearchCV can be computationally expensive, especially with large datasets or a high number of parameters.

### ****Conclusion:****

This analysis shows that both Random Forest and XGBoost are highly effective for classifying mob decisions in the Minecraft dataset. These models achieve high accuracy, precision, recall, and F1 scores, while Logistic Regression, despite performing well, lags behind in comparison. Hyperparameter optimization and model evaluation through cross-validation ensure the robustness and reliability of the results.