Machine Learning Analysis for Predicting Board Game Attributes

Conducted by:

Hassan Almosa

October 19, 2024

Table of Contents

1 Introduction

1.1 Background

Machine learning (ML) provides us with powerful statistical tools to analyse and uncover relationships within large and complex datasets. By leveraging these tools, we can infer connections, identify patterns, and make predictions about various attributes of the data. In this machine learning practice, we are supplied with various board games dataset composed of large amount of features.

1.2 Dataset Description

This practice study utilises a dataset sourced from the Ludii general game system, which encompasses a diverse range of board games characterised by numerous features derived from ludeme keywords. The Ludeme keywords are components that describe the rules, mechanics, and components of a game within the Ludii game system framework.
The dataset comprises 403 instances, each representing a unique board game, with 386 features detailing various aspects of the games. Five target variables (labels) —Category, Region, OriginYear, BestAgent, and UCT—are embedded within the dataset, each requiring independent prediction through our implemented ML models.

1.3 Objectives

The objective of this practice is to apply a thorough ML analysis and techniques to predict these labels by training and evaluating a range of various regression and classification models and techniques. The analysis includes data preprocessing, data exploration, feature engineering, model training, hyper-parameter tuning, and performance evaluation to ensure robust and reliable predictions. The primary objectives of this study are as follows:

- **Regression Tasks:**
  - Predict the *OriginYear*—a numerical value indicating the year a game was first recorded. The years range between negative and positive values representing BCE and CE years, e.g.(-3000, 2024)
  - Predict the *UCT*—a numerical metric representing the average win-rate of a simple UCT-based AI game-playing agents.
- **Classification Tasks:**
  - Classify the *Category*—a categorical label providing a high-level taxonomy of games.
  - Classify the *Region*—a categorical label indicating the geographical origin of the game.
  - Classify the *BestAgent*—a categorical label representing the optimal AI agent currently available for playing the game.

2 Data Preprocessing

2.1 Handling Missing Values

During the initial exploratory data analysis, several columns were identified to have missing values:

- *int_int*: Had 389 out of 403 missing values, representing a high missing ratio thus it was determined its best to be dropped rather than imputed.
- *Board_game.equipment.container.board.Board*: Contains 1 missing value.
- *Equipment_game.equipment.Equipment*: Contains 1 missing value.
- *To_game.functions.ints.iterator.To*: Contains 3 missing values.

2.2 Dropping 'GameName' Column

The GameName column was dropped from the analysis as its contextual value for this task did not exhibit a meaningful relationship with the numerical data. Encoding this feature was considered but would have introduced unnecessary complexity, thus we decided to eliminate the feature given its limited relevance to the current regression and classification objectives with the scope of used ML models that did not include Artificial Neural Networks (ANNs) models capable of such machine learning.

2.3 Imputation Methods

Regression Tasks: Evaluated Mean, Median, and K-Nearest Neighbour (KNN) Imputation. Iterative Imputer was also considered and trialed, but eventually was excluded due to yielding poor simple accuracy metric results and due to its experimental nature.

Classification Tasks: Employed the Most Frequent strategy using SimpleImputer.

2.4 Encoding Categorical Variables

LabelEncoder() was used to convert categorical label variables into numerical form *(Category ,Region, BestAgent).* LabelEncoder was chosen over One-Hot Encoding due to the high dimensionality of the data, which would have been impractical resulting in large number of features.

2.5 Feature Selection/Dimensionality Reduction

Applied Principal Component Analysis (PCA) was preformed on the features set to reduce the feature set high dimensionality (379 features post imputation and exclusion), while retaining 95% variance (n_components=0.95), resulting in 98 principal components which facilitates more efficient computation and enhance the model's performance while retaining dataset's variance.

2.6 Standardisation

Used *StandardScaler()* to normalise feature distributions, and to ensure that all features are scaled and contribute equally to models training process.

2.7 Outlier Detection and Removal

K-Means clustering and Silhouette scores in hand with PCA and t-SNE analysis was employed with to identify outliers. The threshold for removal was samples with Silhouette scores below 0.2 were considered outliers and were removed.

2.8 Train-Test Splits

Performed separate train_test_splits for each label variable to maintain data integrity through out various stages of data processing. Proportion 80% training, 20% testing, with random state set to 42, and stratification (except for 'Region' label in later stage of processing due to have low count of samples) was maintained across the splits.

3 Exploratory Data Analysis (EDA)

3.1 Statistical Summaries

Numerical features (UCT and OriginYear), have shown varying degrees of variance, with OriginYear having a broad range (-3300 to 2019), while UCT was more concentrated.
Categorical features(Category, Region, BestAgent), had issues such as class imbalances, particularly in the Region and BestAgent labels which influenced model performance during classification.

3.2 Visualisations

3.2.1 PCA Plots

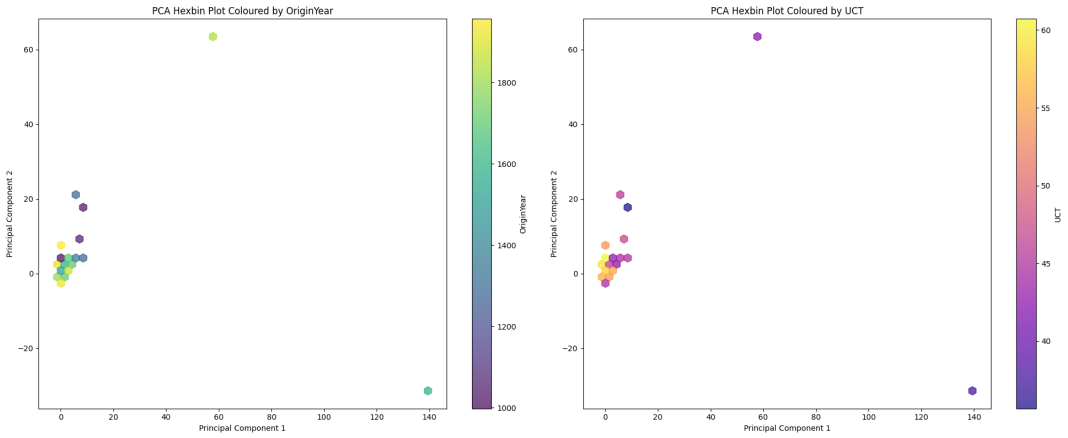

Figure 1: PCA Hexbin Plots for Numerical Labels (OriginYear and UCT)

Note: For a corresponding numerical labels PCA scatter plot, please refer to the Appendix.

3.2.2 t-SNE Plots

To enhance visualisation, t-SNE plots were created using both scatter plots and hexbin heatmaps. The t-SNE process was applied in two stages:

- **Before PCA Reduction**: Imputed, scaled features (see Appendix for numerical labels plot)
- **After PCA Reduction**: Displayed here for categorical and numerical labels (separate plots below)

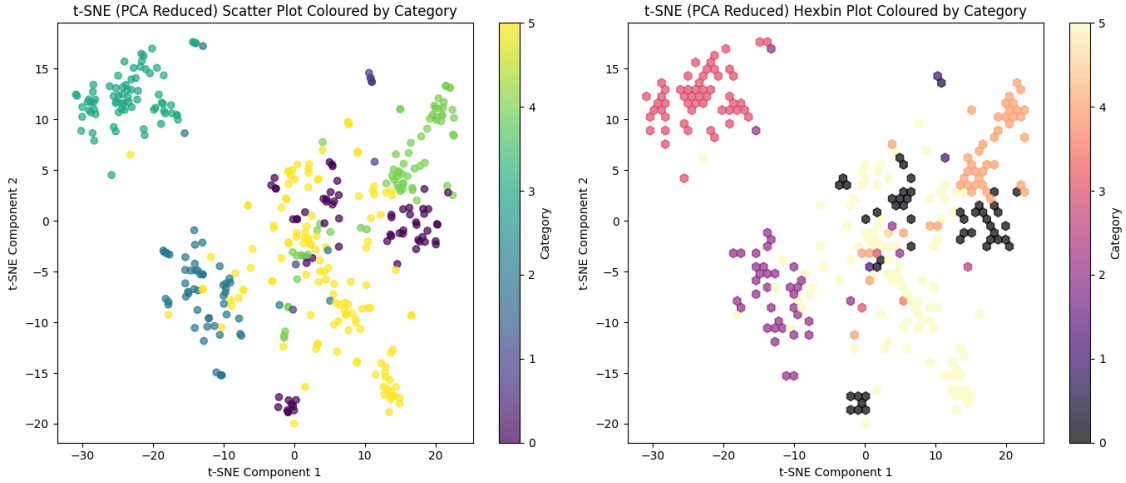

Figure 2: t-SNE Hexbin Plot for Categorical Label (Category) After PCA Reduction

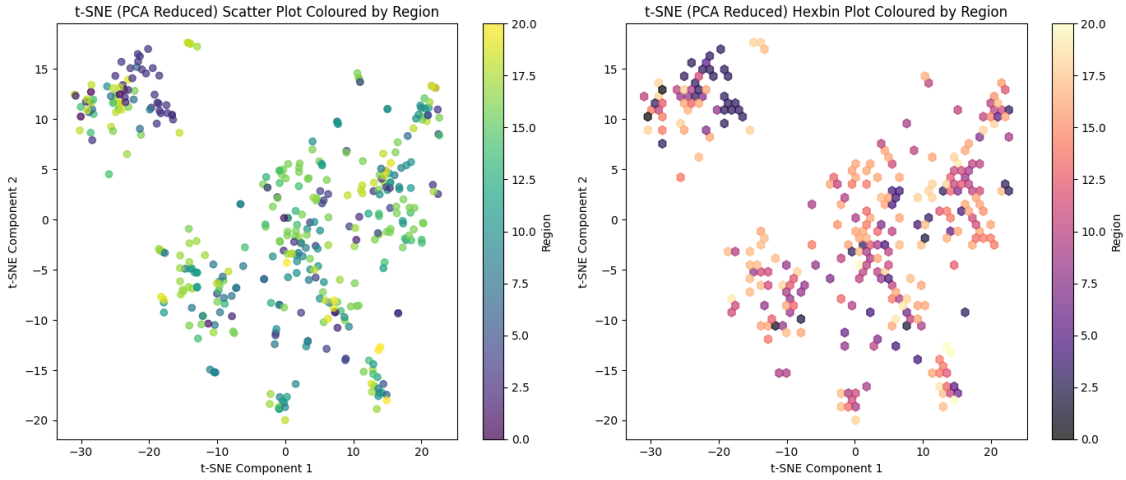

Figure 3: t-SNE Hexbin Plot for Categorical Label (Region) After PCA Reduction

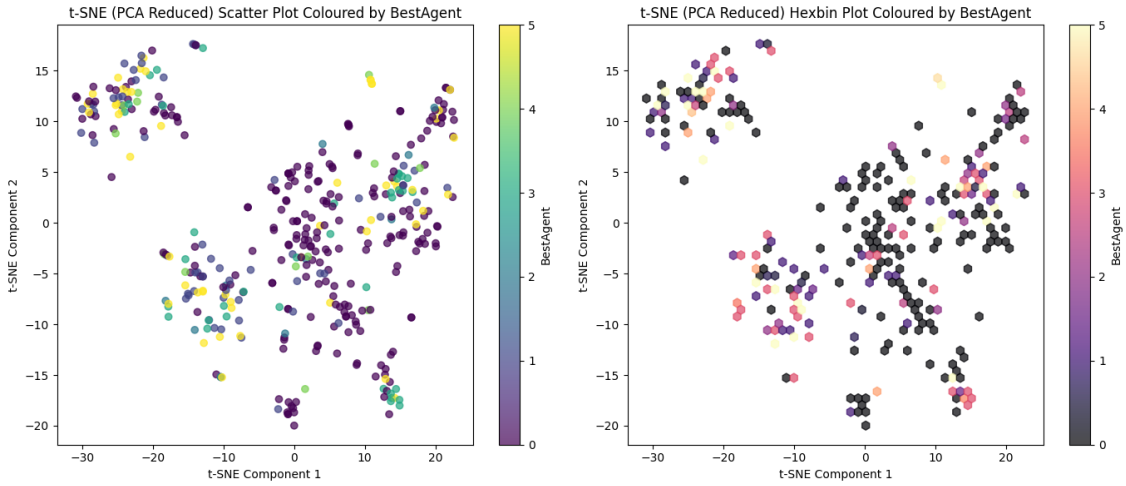

Figure 4: t-SNE Hexbin Plot for Categorical Label (BestAgent) After PCA Reduction

4 Feature Engineering

4.1 K-Means Clustering Evaluation

K-Means Clustering was employed to identify groupings within the PCA reduced feature matrix, and to determine threshold for outliers removal.

- Elbow Method (Inertia vs. Number of Clusters): Determined the optimal number of clusters (k=2) where inertia started to plateau.
- Silhouette Scores: Highest score at k=2 indicating well-defined clusters. Adopted k=2 as the optimal number of clusters for further analysis.

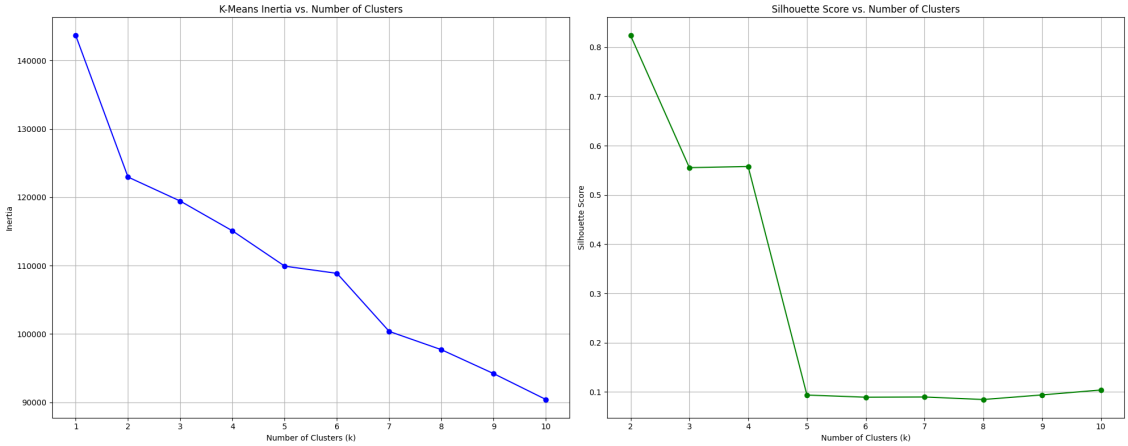

Figure 5: K-Means Inertia vs Number of Clusters Plot (Right) Silhouette Score vs Number of Clusters Plot (Left)

4.2 Cluster Labels as Features

In addition to dimensionality reduction (capturing 95 % variance) K-Means cluster assignments were added into the feature set as additional categorical features (Cluster), this is based on the premise that captured strong clustering relationship infers data patterns and relationships not explictly encoded by the initial ludeme keyword features in the dataset.

5 Results and Discussion

5.1 Summary Tables

Cross-validation testing was conducted twice on both prediction tasks (Regression and Classification), once on validation set created from train_test_split (20% size of the original dataset), and once again lastly using provided testing set of unseen data to our models *(GameData_Prediction.csv)*.

5.1.1 Regression Results

Table 1: Cross-Validation Results for UCT and OriginYear On Validation Set

Model	Target	Mean Squared Error	Mean Absolute Error	R^2
Random Forest	UCT	134.43	8.21	0.27
Stacking Regressor	UCT	123.57	7.98	0.32
Random Forest	OriginYear	438,282.70	277.71	-1.82
Stacking Regressor	OriginYear	124.26	8.05	0.32

Table 2: Cross-Validation Results for UCT and OriginYear On Unseen Test Set

Model	Target	Mean Squared Error	Mean Absolute Error	R^2
Random Forest	UCT	136.30	8.22	0.26
Stacking Regressor	UCT	136.59	8.31	0.25
Random Forest	OriginYear	433,041.03	271.10	-1.86
Stacking Regressor	OriginYear	414,719.31	253.02	-0.74

Both Random Forest and Stacking Regressor struggled with low R^2 scores, suggesting poor fit to the data.

5.1.2 Classification Results

Table 3: Classification Results - Stacking Ensemble

Label	Accuracy	Precision	Recall	F1-Score
Category	0.938	0.930	0.938	0.932
Region	0.370	0.392	0.370	0.365
BestAgent	0.494	0.437	0.494	0.463

Confusion Matrices and Detailed Classification Reports: Detailed confusion matrices and classification reports are presented in the Appendix.
Note: Classification report results (Accuracy, Percision, Recall, F-Score) for our Random Forest Classifier were excluded due to error in generating them and lack of computational resources to re-run the models.

Table 4: Final Cross-Validation Testing on Unseen Test set and Validation set

Model	**Label**	**Test Set CV**	**Val Set CV**
Random Forest Classifier	Category	96.02 %	95.51 %
Stacking Ensemble	Category	93.80 %	94.02 %
Random Forest Classifier	Region	96.27 %	41.90 %
Stacking Ensemble	Region	93.80 %	37.40 %
Random Forest Classifier	BestAgent	96.02 %	59.36 %
Stacking Ensemble	BestAgent	93.80 %	53.87 %

5.2 Best Model Selection

Regression Tasks: Both Random Forest and Stacking Regressor models demonstrated comparable performances with negative R^2 scores, indicating poor predictive capability for OriginYear and UCT.

Classification Tasks: The Stacking Ensemble model and the Random Forest classifier had comparable performance overall, initial prediction runs during early phases of grid search cross-validation yielded poorer results, the results were further enhanced with refining the range of hyperparamters used and the rang of base models employed within the ensemble models to reach the current predictions accuracies. Both models performed poorly on the validation set, with accuracies around 37-42%. However, the accuracies significantly improved on the unseen set. The discrepancy suggests procedural or coding bugs could be the cause and is likely due to class imbalances and overlapping features. This discrepancy becomes more apparent in the classification report for the Region and BestAgent labels in Table 3. Therefore, we cannot conclusively rely on these results for these labels. On the other hand, for the Category label, the cross-validation accuracy on both the validation and test sets supports high accuracy predictions. This is further demonstrated in the classification report on the validation set, showing an F1-score of < 0.9, indicating a high ratio of true positives to the sum of true positives and false positives, reflecting a high degree of precision in the model's predictions.

5.3 Discussion

The dataset was processed and handled with the python Scikit-learn library and performed in Google Collab Pro platform for offering enhanced features and version control, and utility of high-end computational resources. The data was processed initially on Nvidia Tesla A100 GPU to run wide-range grid search cross-validation hyperparameters, and later on various available lower-end TPUs and GPUs, however, with this access, we were still hindered by limitation to access these resources for further efficient re-runs for our models

Both Random Forest Regressor and Stacking Regressor models yielded negative R^2 scores for OriginYear and UCT labels, indicating that the models failed to capture the underlying variance in the data and performed worse than a simple mean predictor. With 379 features of raw game features data, the dataset likely contains some significant noise that might have influenced inaccurate inference and resulted in inability of our regression models to pick up on statistical patterns and correlate meaningfully and correctly. Further PCA deminsoinlaity reduction could possibly enhance the model's learning while further the data engineering process to include more features from studied patterns by clustering techniques and other methods. Moreover, the wide range of negative and positive numbers (-3300 to 2019) may have increased the complexity challenge of our regression models.

For our categorical labels classification tasks, we achieved great accuracy for the Category label, with high F-1 scores indicating effective prediction performance, and with poor to moderate performance on our validation set, but highly accurate prediction on the unseen dataset. This discrepancy begs further examine the models and the code to ensure no bugs were causing this conclusion. Furthermore, an issue of class imbalances was encountered within Region and BestAgent labels which could justify the poor-moderate accuracies. Better techniques for handling class imbalances should be explored and employed for future reproduction. Time, resource, and financial constraints were limiting factors behind further analysis of these discrepancies, and to employ and exploring better techniques to handle them.

6 Conclusion

Machine learning analysis and training was successfully executed for the purpose of this practice task, for both regression and classification tasks to predict various board game attributes, that was achieved by employing data preprocessing methods (imputation and encoding), data visualisation and clustering, outliers handling, dimensionality reduction techniques and features engineeringm training variety of regression and classification models and hyperparameter tuning, and finally evaluation using cross-validation testing. For our classification tasks, models performed exceptionally well for Category predictions but yielded moderate to weak accuracy with Region and BestAgent due to class imbalance and feature limitations. For our regression tasks, the model did exceptionally bad in evaluation metrics, suggesting further need to investigate the used techniques and employing different models such as ANNs and further data engineering methods.

7 References

References

[1] Scikit-learn: Python Machine Learning Documentation. Retrieved from https://scikit-learn.org/

[2] XGBoost Documentation. Retrieved from https://xgboost.readthedocs.io/

[3] Barone, C., Zurowski, S. (2021). Penetration Test Report Template. MIT License. Retrieved from https://github.com/cyber-cfreg/Penetration-Test-Report-Template

[4] Qwen2.5 72B Instruct. (2024). Used for code debugging assistance. Retrieved from https://huggingface.co/Qwen/Qwen2.5-72B-Instruct

[5] Google Colab Pro Gemini. (2024). Auto-completion and code debugging platform. Retrieved from https://colab.google/

[6] Nvidia Llama 3.1 Nemotron 70B Instruct. (2024). Used for LaTeX formatting assistance. Retrieved from https://huggingface.co/nvidia/Llama-3.1-Nemotron-70B-Instruct

8 Appendices

8.1 Additional Results and Code

Please see both attached .ipynb files:

- Numerical_Data_Labels_Predictions[Ludii_Games_Dataset].ipynb
- Categorical_Data_Labels_Predictions[Ludii_Games_Dataset].ipynb

8.2 t-SNE Plots Regression Task Labels

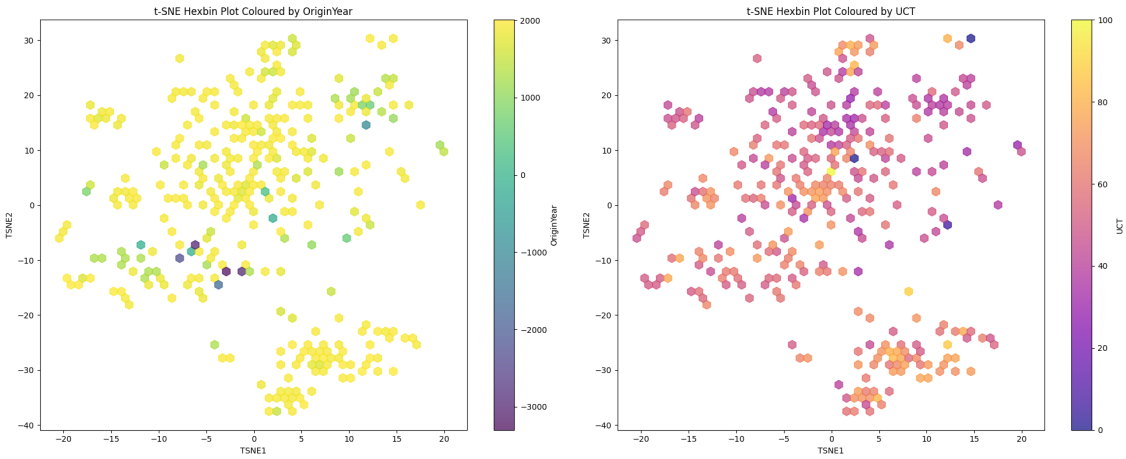

Figure 6: t-SNE Hexbin Plots before applying PCA Coloured by UCT and OriginYear

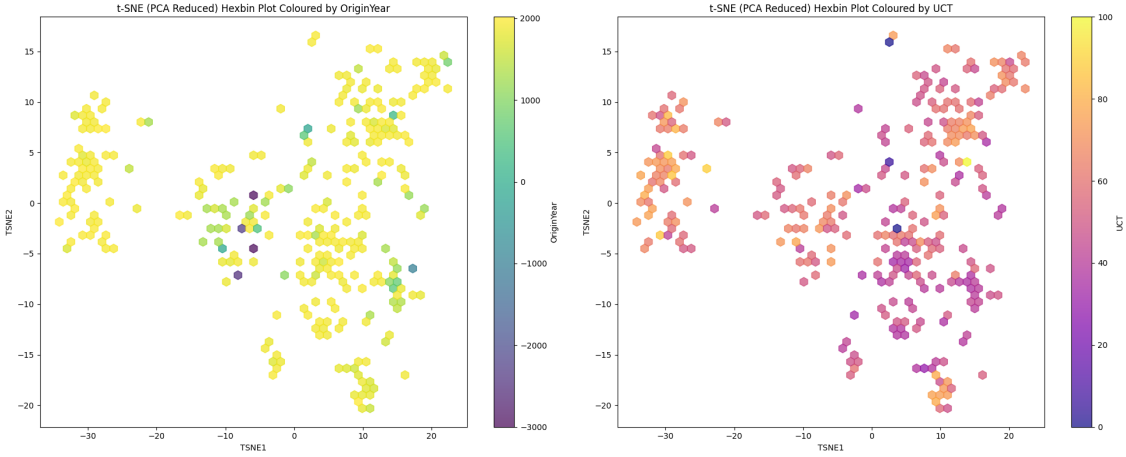

Figure 7: t-SNE Hexbin Plots after applying PCA Coloured by UCT and OriginYear

8.3 PCA Plots For Classification Task Labels

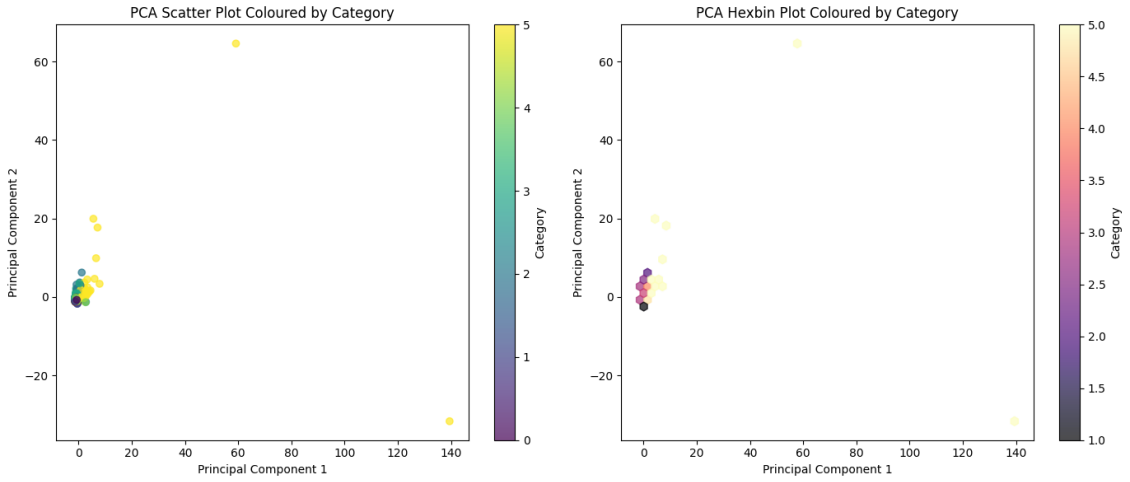

Figure 8: PCA Plots Coloured by Category

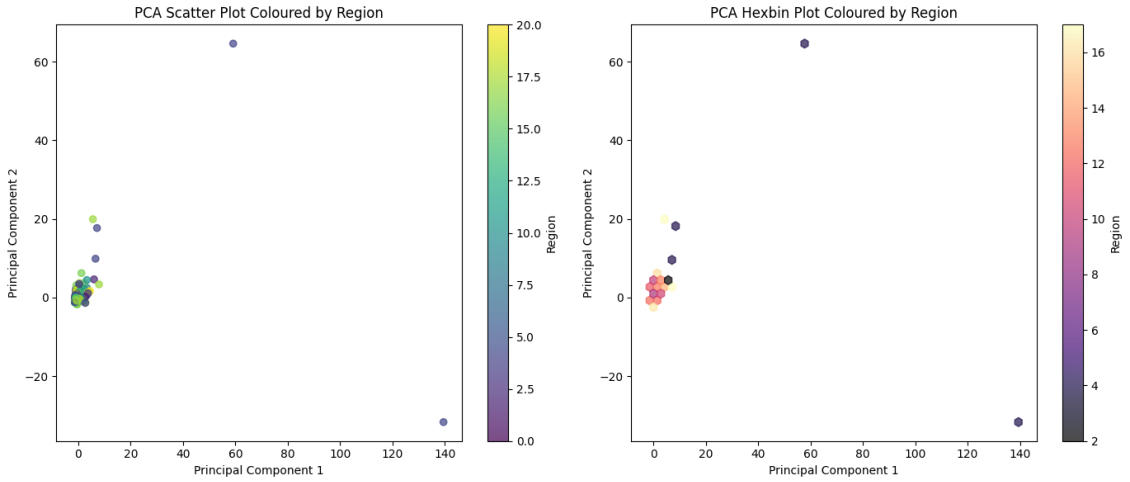

Figure 9: PCA Plots Coloured by Region

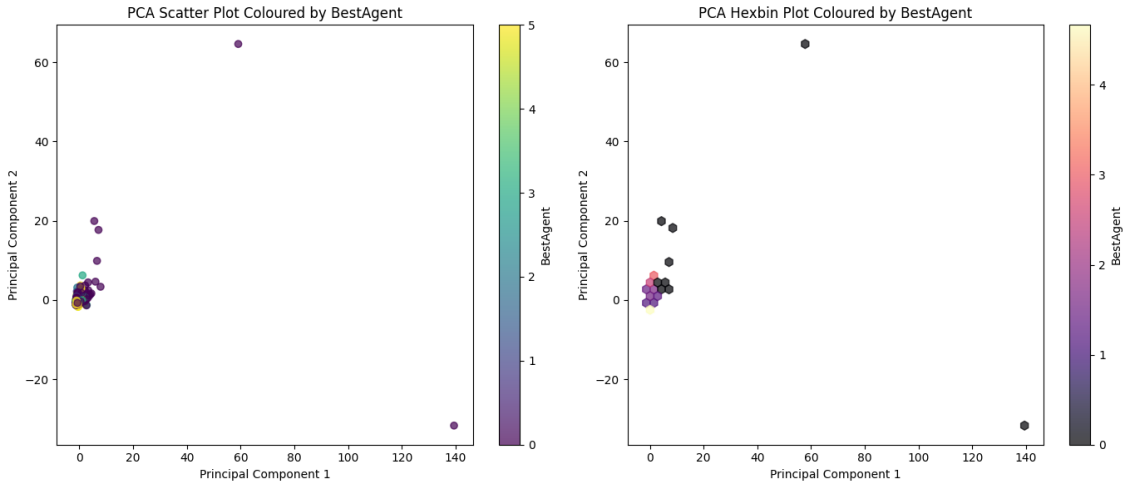

Figure 10: PCA Plots Coloured by BestAgent

8.4 t-SNE Plots Before PCA Application For Classification Task Labels

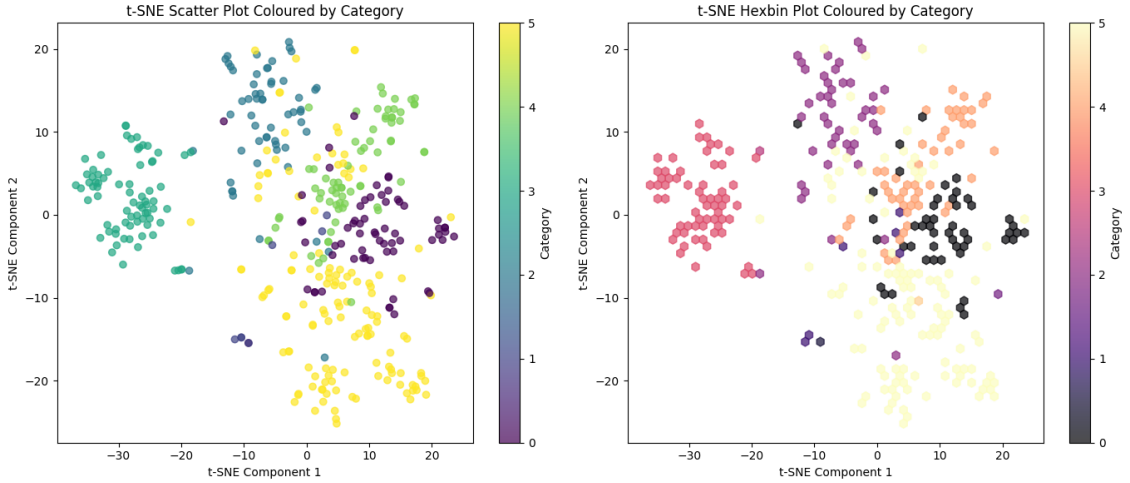

Figure 11: t-SNE plots before applying PCA coloured by Category label

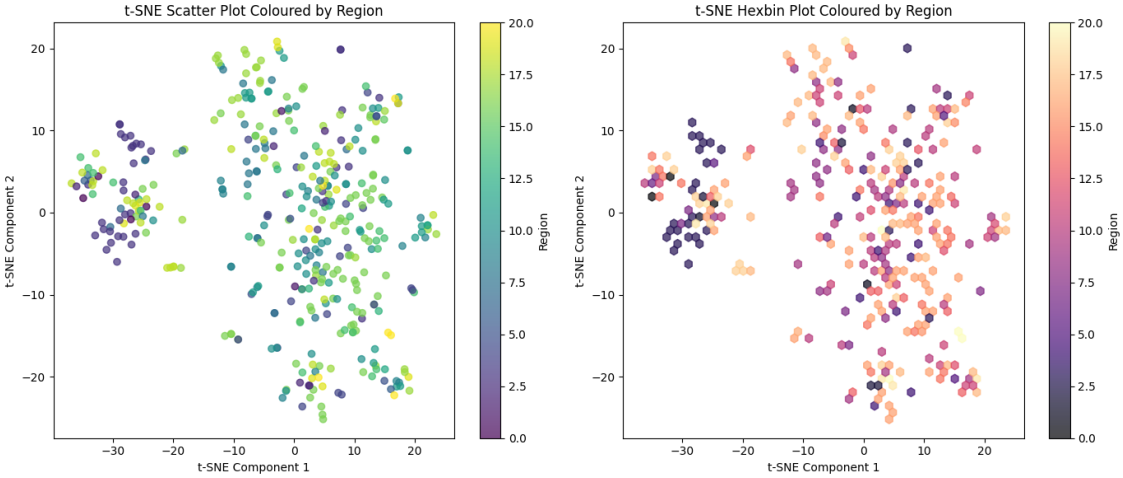

Figure 12: t-SNE plots before applying PCA coloured by Region label

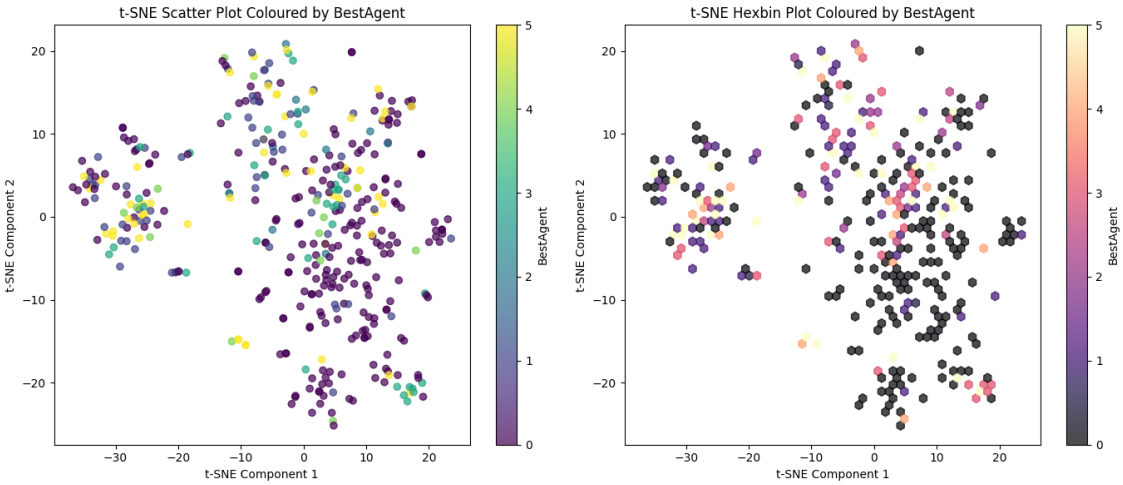

Figure 13: t-SNE plots before applying PCA coloured by BestAgent label

8.5 Confusion Matrices For Classification Task (Random Forrest Model

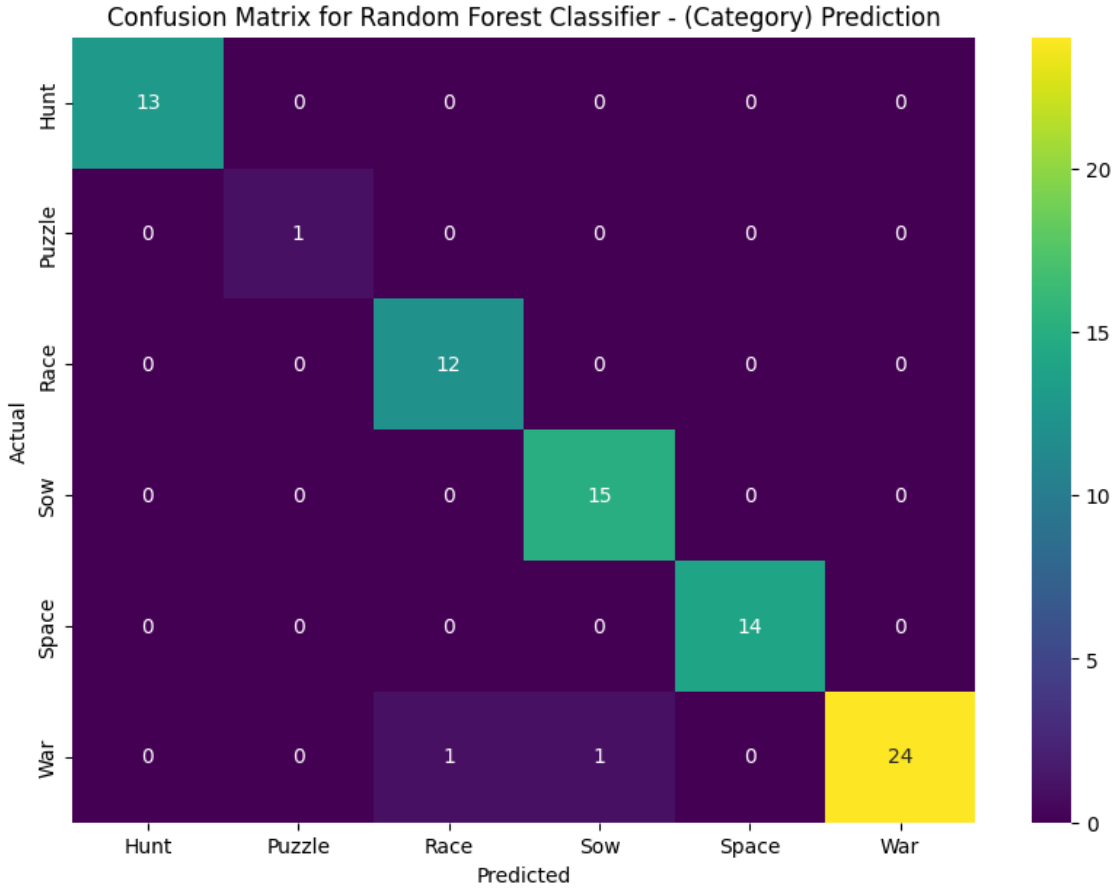

Figure 14: t-SNE plots before applying PCA coloured by Category label

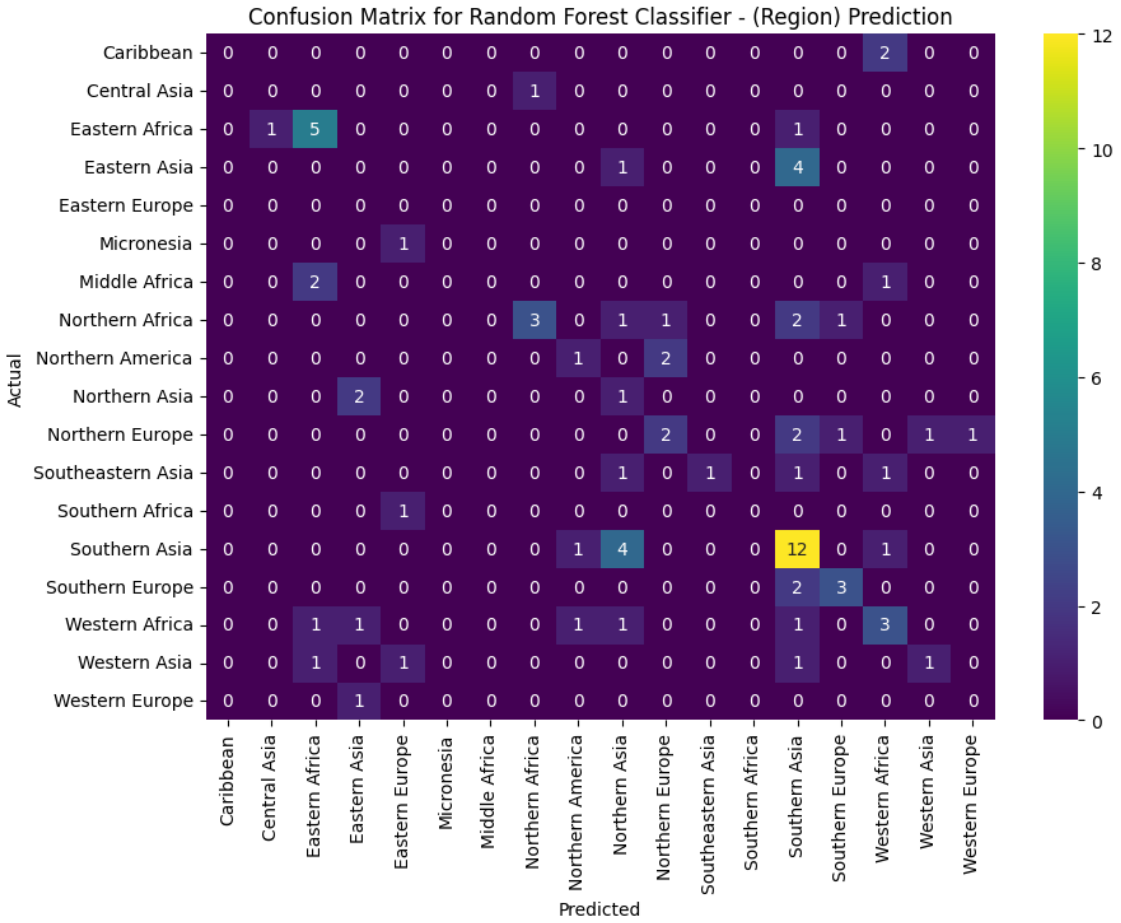

Figure 15: t-SNE plots Before applying PCA coloured by Region label

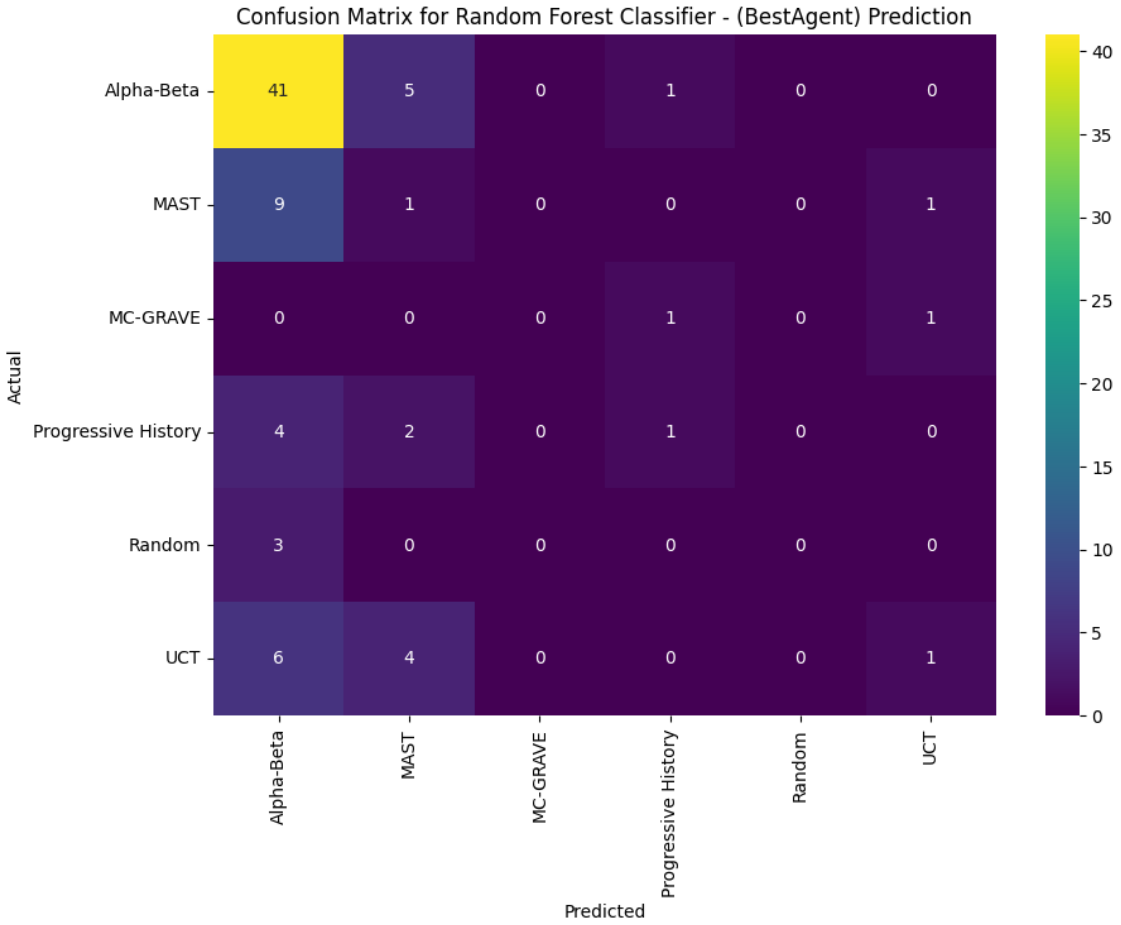

Figure 16: t-SNE plots before applying PCA coloured By BestAgent label

8.6 Confusion Matrices For Classification Task (Stacking Model)

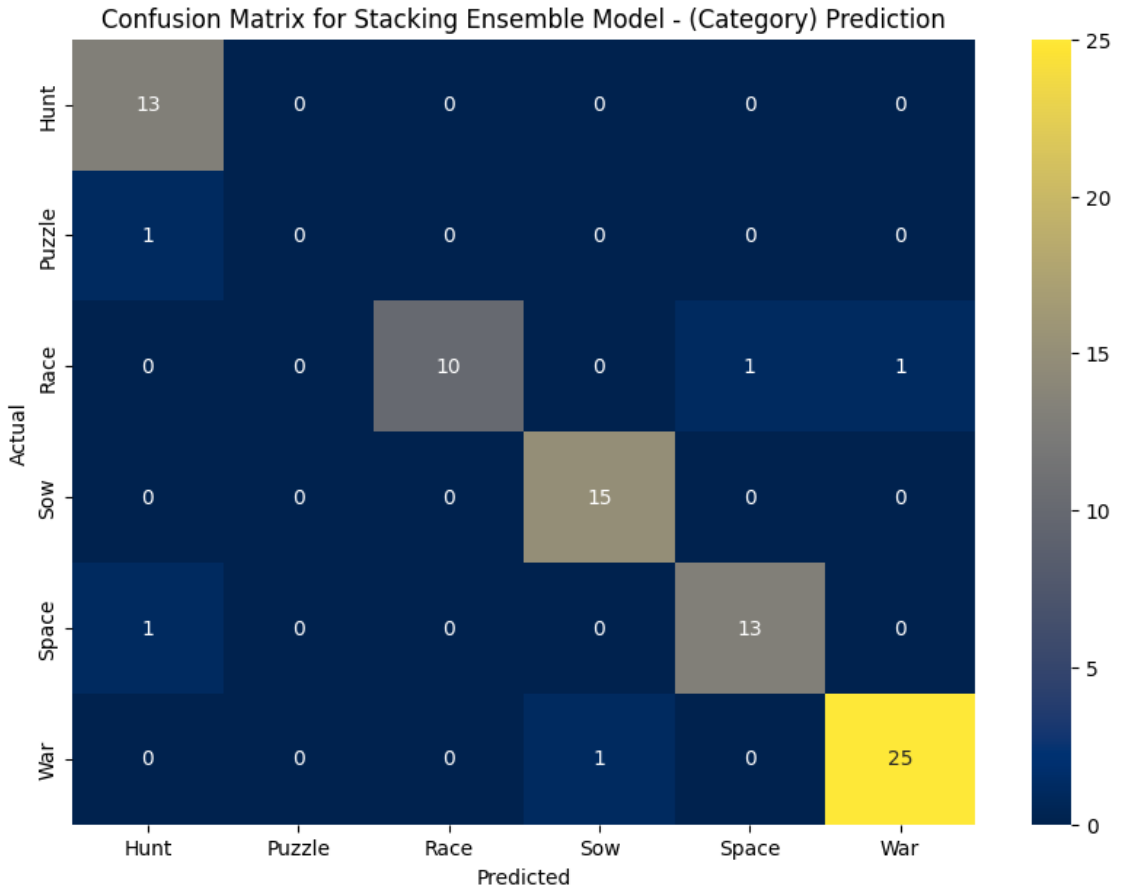

Figure 17: Confusion Matrix for Category label

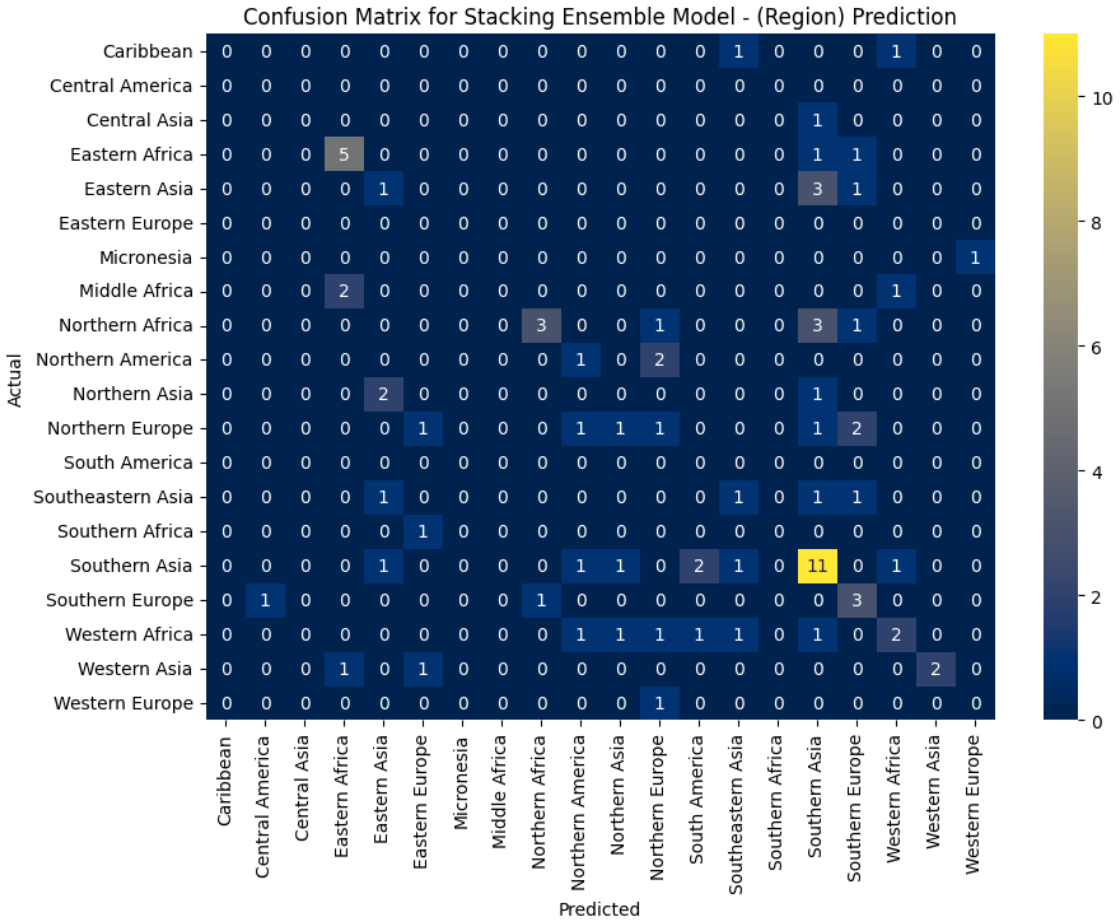

Figure 18: Confusion Matrix for Region label

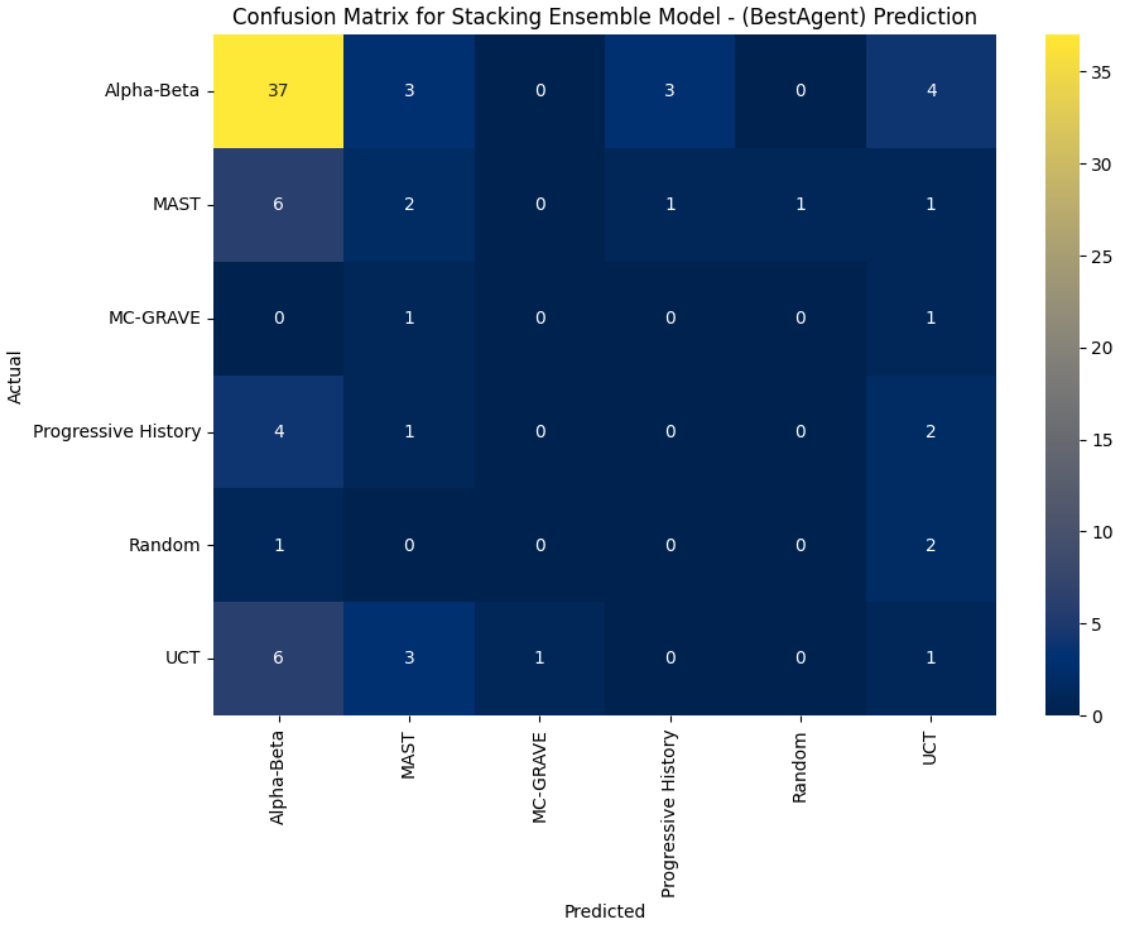

Figure 19: Confusion Matrix for BestAgent label