CZ4041 Machine Learning

A Kaggle Competition Analysis About Predicting Horse Health Outcomes

Submitted by

Xiong Ying

U1920015B

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

NANYANG TECHNOLOGICAL UNIVERSITY

20 Apr 2024

Table of Contents

1. Introduction

This project aims to predict the health outcomes of horses based on a dataset containing various medical indicators, to predict whether the horse can survive based on past medical conditions. The dataset is from the Kaggle playground prediction competition titled "Predict Health Outcomes of Horses (Playground Series - Season 3, Episode 22)", accessible via this link: Kaggle Competition Link.

The primary objective of this project is to excel in the Kaggle competition by achieving a high public score on the leaderboard. Through analysis and classification predictive modelling, this project aims to leverage machine learning techniques to accurately predict the health outcomes of horses based on their medical attributes.

The best performer model in this project achieved a ranking of 101/1543 (top 6.5%) on the Kaggle competition leaderboard, with a public score of 0.84756. Screenshots of the result can be found in **Appendix A: Kaggle Competition Public Score Leaderboard Result**.

Presentation Video Link: YouTube Video Link

Code Repository: GitHub Repository Link

2. Data Exploration and Pre-Processing

This section offers a comprehensive analysis of the dataset, encompassing exploratory data analysis (EDA) and preprocessing steps to facilitate subsequent analysis and modelling.

2.1 Exploratory Data Analysis (EDA)

The dataset comprises synthesized data capturing various medical indicators of horses alongside their corresponding health outcomes. A unified dataset, combining the train and test sets, facilitates thorough analysis to explore the data's characteristics and distributions.

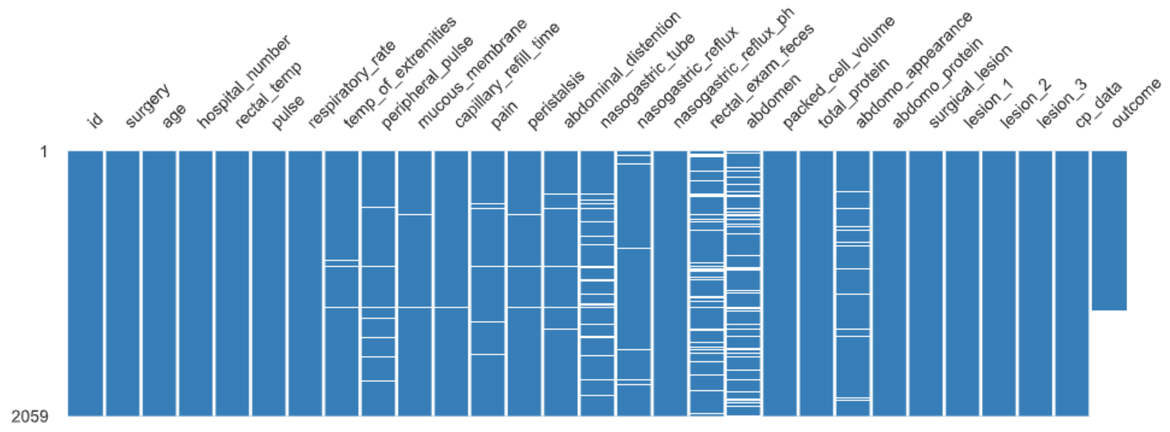
2.2.1 Data Overview

The dataset comprises 29 variables, including physiological measurements like temperature and heart rate, as well as subjective assessments such as pain levels and mucous membrane colour. The dataset consists of 2059 entries, with no duplicate rows.

The variables are categorized into numerical (8 variables), binary (4 variables), and categorical (15 variables) types, with one column for unique identification and one column for the class label.

Approximately 2.3% of the data contains missing values, with some variables exhibiting higher percentages of missing values. The distribution of missing values across the full dataset is visualized using a nullity matrix as illustrated in Figure 1 below.

Figure 1: Nullity Matrix of Missing Values Distribution

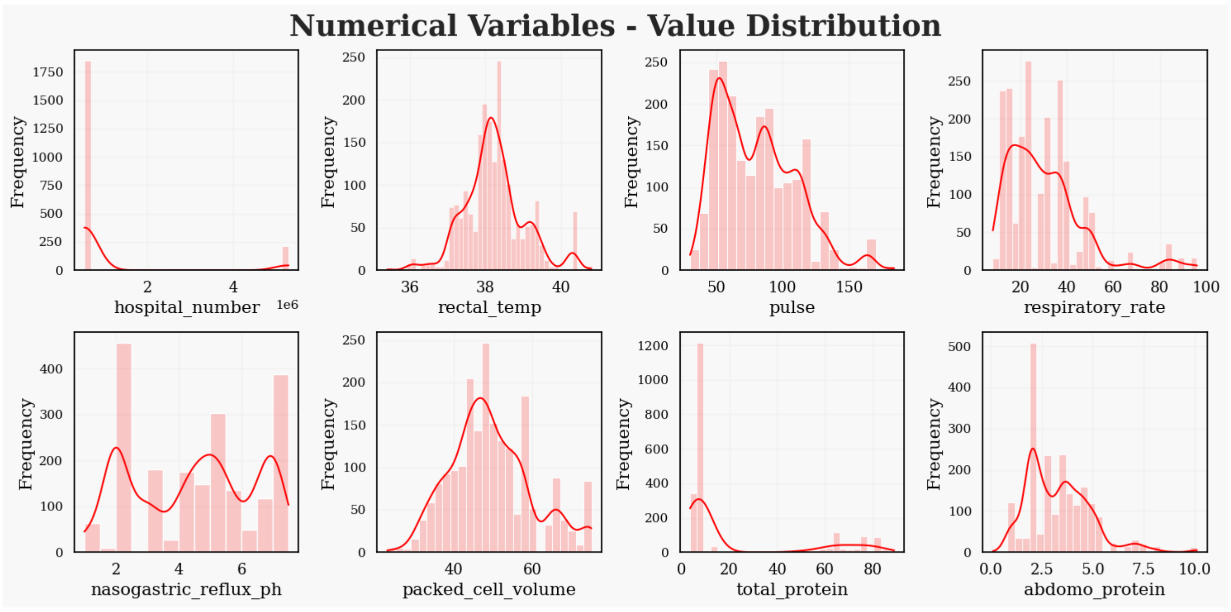


2.2.2 Variable Distribution

1) Numerical Variables

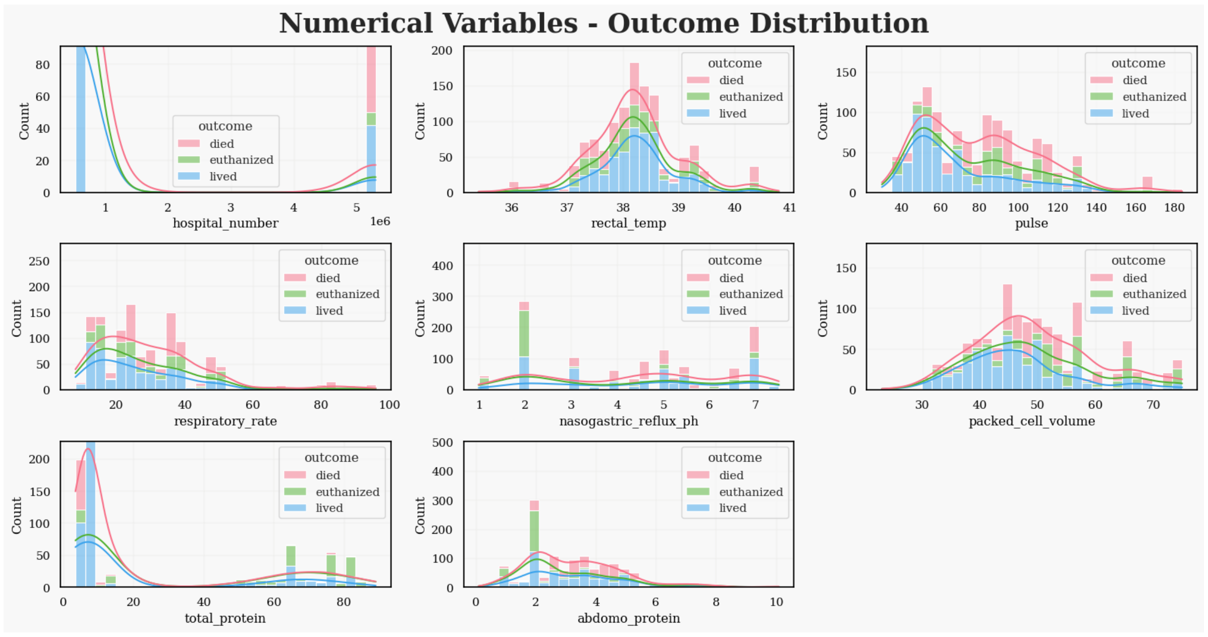
The distribution of unique values for the numerical variables is visualized in Figure 2 below. The numerical variables exhibit varied distributions: `hospital_number` and `total_protein` display imbalances, `rectal_temp` and `packed_cell_volume` show central tendencies, `pulse`, `respiratory_rate`, and `abdomo_protein` are skewed, potentially correlated with `outcome`. `Nasogastric_reflux_ph` fluctuates, and `total_protein` correlates with "died" and "euthanized" outcomes.

Figure 2: Numerical Variables - Value Distribution



Additionally, the count plot (Figure 3) of the target variable `outcome` for each numerical variable provides insights into their potential relationships. `Hospital_number` influences `outcome`, with lower values mostly resulting in "lived" outcomes. `Rectal_temp` and `Packed_cell_volume` show central distributions for different outcome categories. `Pulse`, `Respiratory_rate`, and `Abdomo_protein` display skewed distributions.

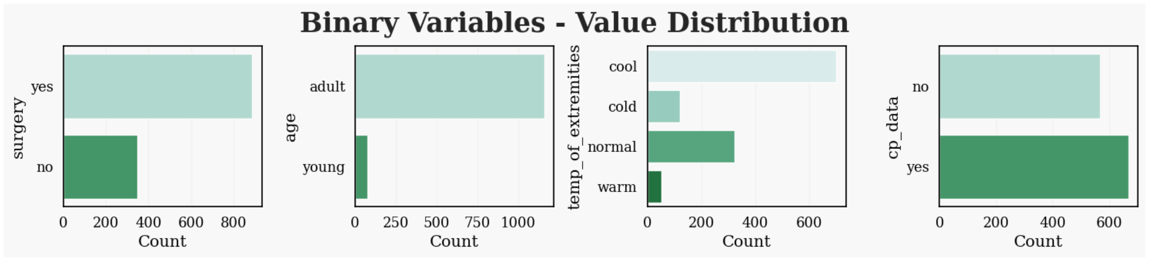
Figure 3: Numerical Variables - Outcome Count Plot



2) Binary Variables

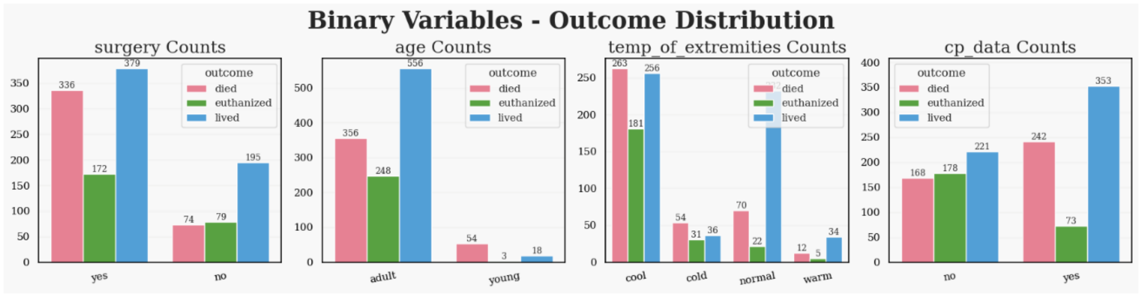
As shown in Figure 4, in the binary variables, there's an imbalance with predominantly adult entries in the **`age`** column and a slight bias towards "yes" entries in the **`surgery`** column, indicating higher surgery frequency. Additionally, **`temp_of_extremities`** lean towards the "cool" category, while **`cp_data`** shows an even distribution among its categories.

Figure 4: Binary Variables - Value Distribution



The count plot (Figure 5 below) demonstrates that not undergoing surgery correlates with higher survival rates. Moreover, horses with normal temperatures in their extremities show a greater likelihood of survival.

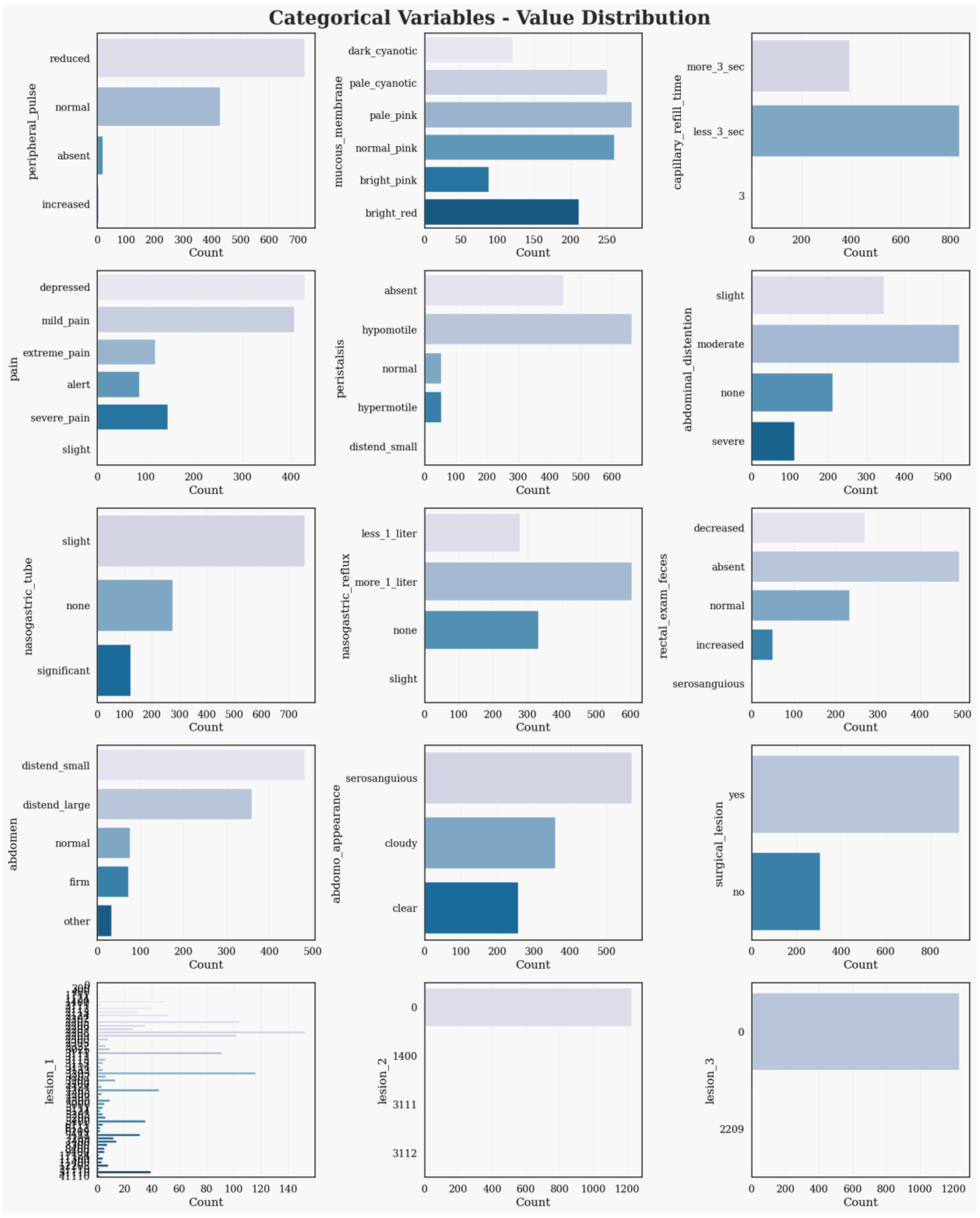
Figure 5: Binary Variables - Outcome Count Plot



3) Categorical Variables

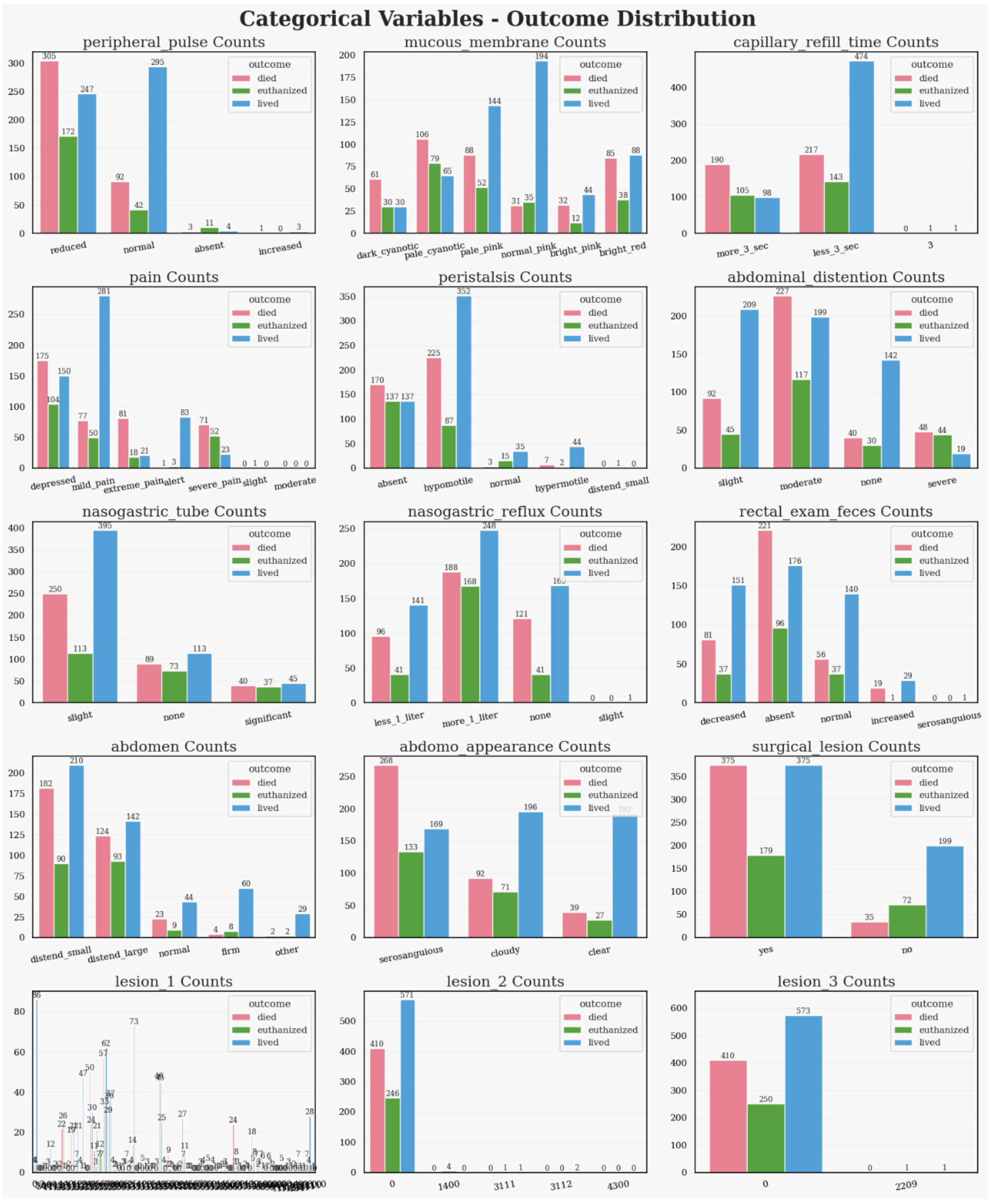
As shown in Figure 6, most categorical variables, like `pain`, `peristalsis`, and `abdominal_distention`, have diverse observations across different levels. Many of these variables display imbalanced distributions, including `peripheral_pulse` and `nasogastric_tube`. Interestingly, columns `lesion_2` and `lesion_3` contain numerous "zero" entries, which may lack meaningful information or relevance.

Figure 6: Categorical Variables - Value Distribution



Certain categorical variables seem to have potential relationships with the outcome variable as illustrated in Figure 7. For example, specific values like "normal" for `peripheral_pulse` or "normal_pink" for `mucous_membrane` appear to correlate with a higher likelihood of a "lived" outcome. Similarly, categories such as "less_3_sec" for `capillary_refill_time` or "mild_pain" for `pain` indicate favourable outcomes. Additionally, the value "no" for `surgical_lesion` also appears to be associated with positive outcomes.

Figure 7: Categorical Variables - Outcome Count Plot

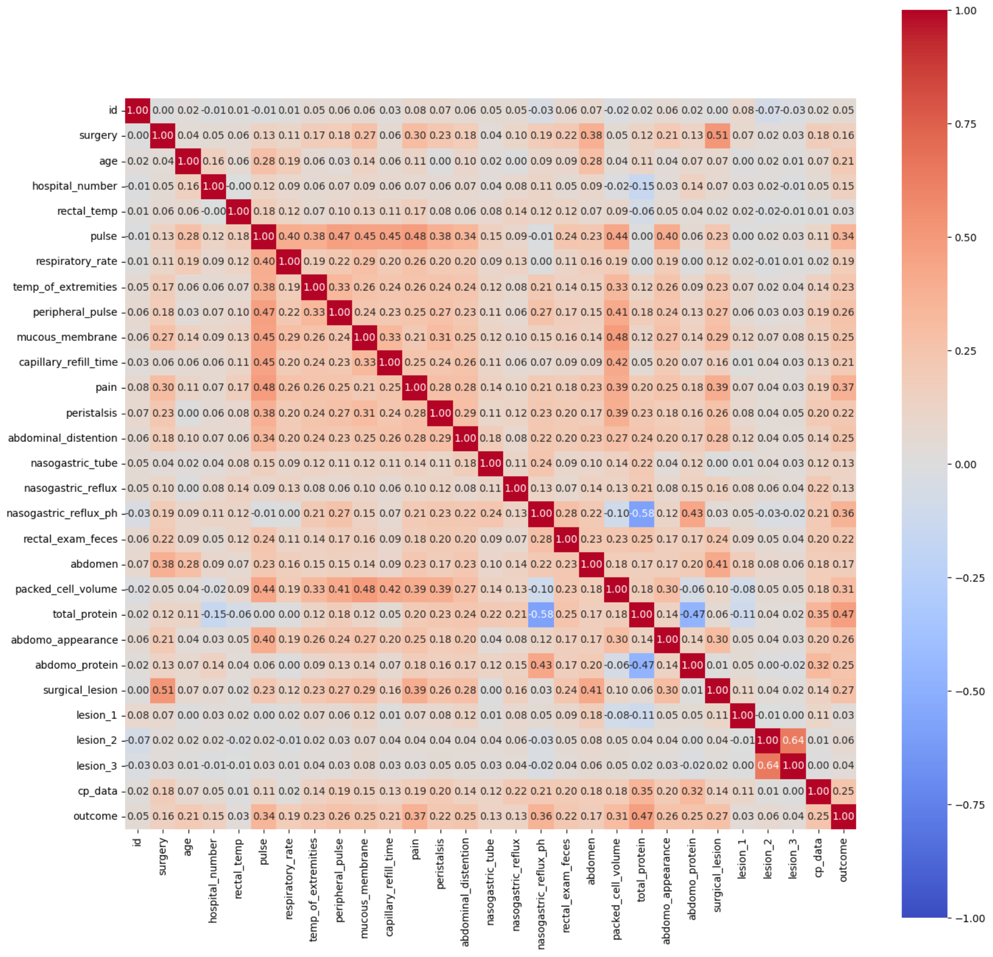


2.2.3 Feature Correlation and Importance

1) Correlation Matrix

A correlation matrix was created to examine relationships between variables, with values representing correlation coefficients between pairs of variables.

Figure 8: Heatmap



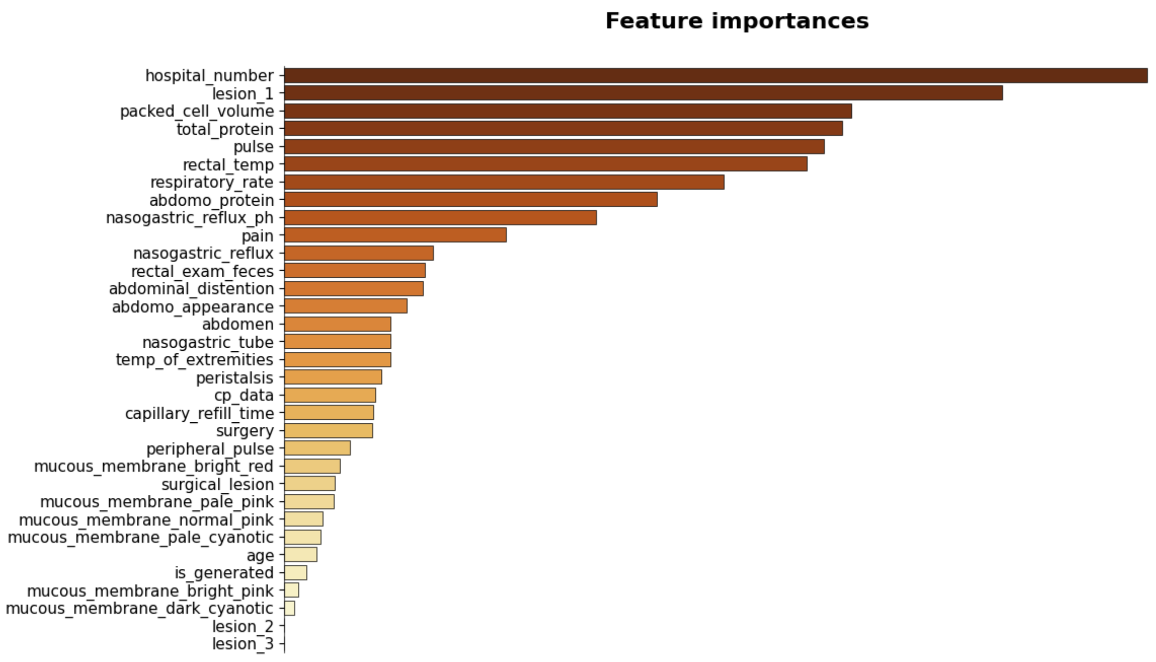
Key observations include:

- Numerous features show low or negligible positive correlations with the "outcome" variable, with no negative correlations observed.
- Prioritizing features with stronger positive correlations with the outcome during model training is suggested, but even weakly correlated features may contain valuable information for predictive performance assessment.
- Physiological measurement-related features display positive correlations around 0.4, indicating potential multicollinearity issues that may require feature selection techniques, such as `pulse`, `respiratory_rate`, `temp_of_extremities`, etc.
- `lesion_2` and `lesion_3` demonstrate a substantial correlation of 0.644, while a correlation of 0.509 exists between `surgery` and `surgical_lesion`.
- Additionally, `total_protein` and `nasogastric_reflux_ph` exhibit a significant negative correlation of -0.583.

2) Feature Importance

Feature importance evaluation helps refine the feature set for improved model performance. Variable `hospital_number` and `lesion_1` are identified as the most influential variables, while `lesion_2` and `lesion_3` are deemed insignificant. Thus, it's suggested to consider removing these variables to enhance relevance.

Figure 9: Feature Importances



2.3 Pre-Processing

The pre-processing phase involves several steps tailored to different variable types:

1) **Handling Missing Values:** Mode imputation fills missing values with the most frequent value in each column.

2) **Normalizing Numerical Variables:** Min-Max normalization scales numerical features between 0 and 1 to mitigate the dominance of large values.

3) **Encoding Binary Variables:** Label encoding maps True to 1 and False to 0.

4) **Encoding Categorical Variables:**

   - **One-Hot Encoding**: Used for one-vs-rest variables like **`mucous_membrane`**.
   - **Label Encoding**: Applied to ordinal variables, mapping values to integers based on their ordinal sequence.

5) **Feature Selection**: Only significant features related to the **`outcome`** variable are retained, removing variables like **`lesion_2`** and **`lesion_3`**.

6) **Other Considerations**:

   - The **`id`** variable is excluded from the feature selection.
   - The **`outcome`** variable is encoded into numerical values (0, 1, 2) for model training.

3. Methodologies

3.1 Model Selection

Various single classifier models and ensemble methods are selected for addressing the classification problem and accessing the performance of each method.

Single classifier models include Support Vector Machine (SVM), K-Nearest Neighbours (KNN), Decision Tree (DT), and Artificial Neural Network (ANN). Ensemble Methods include Bagging, Random Forest (RF), Stacking, AdaBoost, Light Gradient Boosting (LGB), Extreme Gradient Boosting (XGB), Histogram-Based Gradient Boosting (HGB).

These models are readily implementable with the support of established libraries such as **`Scikit-learn`**, **`torch`**, **`LightGBM`**, and **`XGBoost`**. Through experimentation with diverse classifiers, we aim to explore the performance landscape and gain insights into the underlying characteristics of the dataset.

3.2 Cross-Validation

The dataset is divided into train and test sets, with only the training set used for model training. The training set is further split into training and validation sets to evaluate model performance during training. Trained models are then used to predict classes for the test set.

A 5-fold cross-validation approach is employed for all models, dividing the training data into five parts for validation. This technique ensures robustness and reliability in evaluating model performance, especially with unseen data.

3.3 Hyperparameter Tuning

Grid Search was employed for hyperparameter tuning across all models, systematically exploring predefined combinations of hyperparameters to identify the optimal set that maximizes accuracy. The **`GridSearchCV`** function from the **`Scikit-learn`** library was utilized to exhaustively search these hyperparameters using cross-validation to ensure that the models are fine-tuned to achieve optimal performance.

3.4 Evaluation Metrics

The Kaggle competition used the F1-score using the micro-average method as the primary evaluation metric. These metric balance precision and recall by considering True Positives (TP), False Positives (FP), and False Negatives (FN) [1]. The formula for computing this score is provided below. The micro-average method calculates the F1 score across all classes, making it suitable for multi-class classification tasks. The **`ClassificationReport`** module within the **`Scikit-learn`** package was used to compute the F1 micro-average score.

$$MicroF1 = \frac{2 * Precision * Recall}{Precision + Recall}, where\ Precision = \frac{TP}{TP + FP}, Recall = \frac{TP}{TP + FN}$$

4. Experiments

This section explains the rationale for choosing each classifier model and ensemble method and presents the outcomes of hyperparameter tuning. It then showcases the F1 micro-average score, indicating the training set accuracy. Following this, the models were employed to predict outcomes for the test set, and the resulting predictions were submitted to Kaggle for evaluation, obtaining the public leaderboard score, which reflects the test accuracy.

4.1 Single Classifier Models

4.1.1 Support Vector Machine (SVM)

1) Model Selection Rationale: Support Vector Machine (SVM) emerges as the preferred choice due to its ability to manage intricate decision boundaries and handle datasets with nonlinear relationships, which aligns with the dataset's characteristics [2]. It performs well in high-dimensional spaces when data has many features. Furthermore, SVM is known for its efficacy in handling multi-class classification tasks by effectively identifying clear margins of separation between multiple categories, which is likely relevant to the objectives of this project.

However, one drawback of SVM is its performance degradation in datasets with high noise levels or overlapping classes, as it relies on clear margins of separation between classes.

2) Hyperparameter Optimization Results:

Hyperparameter	**Pre-defined Values for Tuning**	**Best Values**
C	0.01, 0.1, 1, 10, 100	1
gamma	1, 0.1, 0.01,0.001,0.0001,0.000001,0.0000001	0.1
kernel	‘linear', 'poly', 'rbf', 'sigmoid’	‘rbf’

3) Evaluation: After optimization, the SVM model achieved an accuracy of 0.70 in the train set, and 0.73170 in the test set, signifying good performance.

4.1.2 K-Nearest Neighbours (KNN)

1) Model Selection Rationale: K-Nearest Neighbours (KNN) is an accessible choice for modelling tasks for its ease of implementation and understanding. Its lazy learning approach, where the model doesn't make assumptions about the underlying data distribution during training, can be advantageous in scenarios where the data distribution is not well-defined or when the relationships between features and the target variable are complex [3].

However, it's important to note some limitations of KNN. High dimensionality can hinder its performance. Additionally, it can be computationally expensive, especially with large datasets, due to its need to calculate distances between each pair of data points. The choice of 'k' and distance metrics can also affect its performance. Furthermore, KNN may struggle with imbalanced classes and missing values, which could affect the accuracy of classifying instances.

2) Hyperparameter Optimization Results:

Hyperparameter	Pre-defined Values for Tuning	Best Values
n_neighbors	2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20	6
weights	'uniform','distance’	'uniform'
metric	'minkowski','euclidean','manhattan’	'minkowski'
p	1,2,4,8,16	1

3) Evaluation: The optimal KNN model achieved a training set accuracy of 0.75 and a test set accuracy of 0.75000, suggesting its capacity to generalize well to unseen data.

4.1.3 Decision Tree (DT)

1) Model Selection Rationale: Decision Tree (DT) is known for its interpretability, which is essential for understanding the decision-making process underlying the predictions [4]. This attribute is particularly valuable in domains where explainability is crucial, such as healthcare or finance. Additionally, DTs are good at capturing non-linear relationships within the data, making them well-suited for datasets with complex patterns or interactions between variables without making strong assumptions about the functional form of the relationships.

However, DTs are prone to overfitting, especially with deep trees or when the dataset contains noisy or irrelevant features. Moreover, DTs can be sensitive to outliers, which may disproportionately influence the structure of the tree and impact performance.

2) Hyperparameter Optimization Results:

Hyperparameter	Pre-defined Values for Tuning	Best Values
criterion	'gini','entropy’	'gini’
max_depth	2,3,4,5,6,7,8,9,10	3
min_samples_split	2,3,4,5,6,7,8,9,10	2
max_leaf_nodes	3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25	5

3) Evaluation: The optimal Decision Tree model achieved a training set accuracy of 0.68 and a test set accuracy of 0.72560, suggesting its capacity to generalize well to unseen data.

4.1.4 Artificial Neural Network (ANN)

1) Model Selection Rationale: Artificial Neural Network (ANN) stands out for its ability to capture intricate patterns through multiple layers of neurons, making them suitable for datasets with complex relationships between variables [5]. One of the main advantages of neural networks is their ability to implicitly detect complex nonlinear relationships between dependent and independent variables. Furthermore, ANNs require less formal statistical training compared to traditional statistical models. They also provide multiple training algorithms, allowing for customization based on the specific characteristics of the dataset.

However, the "black box" nature of ANNs means that the internal mechanisms of the model are not easily interpretable, which can make it challenging to understand the reasoning behind predictions. Additionally, ANNs typically require greater computational resources compared to some other models.

2) Hyperparameter Optimization Results:

Hyperparameter	Pre-defined Values for Tuning	Best Values
hidden_dim	8, 16, 24, 32	24
hidden_layers	1, 2, 3, 4, 5	4
lr	0.1, 0.01, 0.001	0.001
num_epochs	50, 100, 150, 200, 250, 300	150

3) Evaluation: The optimal ANN model achieved a training set accuracy of 0.67 and a test accuracy of 0.65853. Both scores are slightly lower than the previous classifiers.

4.2 Ensemble Methods

4.2.1 Bagging

1) Model Selection Rationale: Bagging, or Bootstrap Aggregating, is chosen as it reduces variance by training multiple learners on slightly different data subsets of the same size on the same algorithm and averaging their predictions, leading to diverse models that collectively provide a more robust prediction, enhances stability and generalization by mitigating the impact of outliers and noise [6].

However, Bagging may not be as effective in addressing bias or underfitting in the data, as its primary focus is on reducing variance. Additionally, Bagging ignores extreme values by averaging the predictions of multiple models, which may lead to a loss of information.

2) Hyperparameter Optimization Results:

Hyperparameter	Pre-defined Values for Tuning	Best Values
n_estimators	20,40,80,160,320	20
max_samples	0.5,0.8,1	0.5
max_features	0.5,0.8,1	0.5
bootstrap	True, False	True

3) Evaluation: The optimal Bagging model achieved a train accuracy of 0.49 and a test accuracy of 0.46341, indicating moderate performance. This approach performs less effectively compared to the single classifier because it incorporates fewer entries for training each time, and the base estimator SVM does not perform optimally with reduced features.

4.2.2 Random Forest (RF)

1) Model Selection Rationale: Random Forest (RF) is an ensemble learning method that combines the predictions of multiple decision trees to reduce overfitting and improve generalization [7]. This involves training each decision tree on a different subset of the data with a random set of features, which helps to prevent any single decision tree from dominating the ensemble, thereby reducing the risk of overfitting. Additionally, RF can achieve higher accuracy by leveraging the collective result of multiple decision trees.

However, RF has higher complexity, as RF involves training multiple decision trees and aggregating their predictions. Moreover, RF models may not be as interpretable, making it more challenging to understand the underlying decision-making process.

2) Hyperparameter Optimization Results:

Hyperparameter	Pre-defined Values for Tuning	Best Values
n_estimators	20,40,80,160,320	160
max_features	'auto', 'sqrt'	'sqrt'
max_depth	3,6,12,24	12
min_samples_split	2, 5, 10	2
min_samples_leaf	1, 2, 4	1
bootstrap	True, False	False
criterion	'gini', 'entropy', 'log_loss'	'gini'

3) Evaluation: The optimal Random Forest model achieved a train accuracy of 0.99 and a test accuracy of 0.77439, indicating strong performance in both training and generalization.

4.2.3 Stacking

1) Model Selection Rationale: Stacking is chosen for its ability to improve predictive performance by using a meta-learner to make predictions based on the results from multiple models, leveraging the diverse perspectives of individual models, aiming to make more accurate predictions compared to each model [8].

However, one significant drawback is the increased complexity it introduces, making the final model harder to explain. Additionally, Stacking may incur higher computation time, especially when dealing with large volumes of data. The increased complexity of the model can lead to longer processing times. Furthermore, the effectiveness of Stacking depends on the diversity of the base models used. It is most effective when combining models that are not highly correlated, as this diversity provides more opportunities to optimize the performance.

2) Hyperparameter Optimization Results:

Hyperparameter	Pre-defined Values for Tuning	Best Values
final_estimator	svm.SVC(), KNeighborsClassifier(), DecisionTreeClassifier(), BaggingClassifier(), AdaBoostClassifier(), RandomForestClassifier(), LogisticRegression()	LogisticRegression()
passthrough	True, False	False
stack_method	'auto', 'predict_proba', 'decision_function', 'predict'	auto

3) Evaluation: The optimal Stacking model achieved a train accuracy of 1.00 and a test accuracy of 0.78658. By leveraging the strengths of multiple base models and learning from their combined predictions, the Stacking ensemble demonstrates great predictive capability.

4.2.4 AdaBoost

1) Model Selection Rationale: AdaBoost employs an iterative approach to train weak learners sequentially, which focuses on instances that were misclassified by previous models, ultimately enhancing the overall performance of the model [9]. Another advantage of AdaBoost is its immunity from overfitting, as it runs each model in a sequence and assigns weights to them based on their performance. This helps prevent the model from memorizing the training data. Additionally, AdaBoost requires minimal hyperparameter tuning.

However, it's important to note that AdaBoost requires quality data for training, as it is sensitive to noisy data and outliers, it may mistakenly boost the weights of noisy instances, leading to suboptimal performance. Besides, the performance of AdaBoost heavily depends on the quality and the base learners used in the ensemble. If the base learners are weak, AdaBoost may not achieve significant performance improvements.

2) Hyperparameter Optimization Results:

Hyperparameter	**Pre-defined Values for Tuning**	**Best Values**
n_estimators	20,40,80,160,320	80
learning_rate	0.001,0.01,0.1,0.3,0.5,0.7,0.9	0.9

3) Evaluation: The optimal AdaBoost model achieved a train accuracy of 0.75 and a test accuracy of 0.76829, indicating reasonable performance. This method shows improvement over the Bagging approach and yields better results.

4.2.5 Light Gradient Boosting (LGB)

1) Model Selection Rationale: LGB is chosen for its utilization of gradient boosting techniques, which iteratively enhance the efficacy of weak learners in a gradient descent manner, to minimize the bias error of the model and result in highly precise predictions [10]. One of the main advantages of LGB is its ability to handle large amounts of data efficiently, making it computationally efficient with fast training speeds and low memory usage.

However, LGB can potentially overfit due to leaf-wise splitting and high sensitivity to hyperparameters, which can make hyperparameter tuning complex. Finding the optimal combination of hyperparameters may require extensive tuning. This process can be time-consuming and computationally expensive, particularly for datasets with many features or observations.

2) Hyperparameter Optimization Results:

Hyperparameter	**Pre-defined Values for Tuning**	**Best Values**
reg_alpha	0.01,0.05,0.1	0.05
reg_lambda	0.01,0.05,0.1	0.01
learning_rate	0.001,0.01,0.1,0.3,0.5,0.7,0.9	0.1
max_depth	3,6,12	3
n_estimators	50,100,200	200

3) Evaluation: The optimal LGB model achieved a train accuracy of 0.93 and a test accuracy of 0.81707. The high test accuracy indicates that the model generalizes well to unseen data, confirming its effectiveness in making accurate predictions.

4.2.6 Extreme Gradient Boosting (XGB)

1) Model Selection Rationale: Extreme Gradient Boosting, also known as XGBoost (XGB) is a gradient boosting framework known for its comprehensive capabilities and expanded array of hyperparameters, which allows for fine-tuning the model to better fit the dataset [11]. By iteratively improving the model's predictions using weak learners, XGB can capture complex, nonlinear relationships present in the data.

However, the potential drawback of using XGB is that hyperparameter tuning can be complex due to the large number of hyperparameters available, which may require significant computational resources and time for optimization. The ensemble nature of XGBoost makes it difficult to understand the individual decision-making process of each tree, limiting the model's interpretability.

2) Hyperparameter Optimization Results:

Hyperparameter	Pre-defined Values for Tuning	Best Values
alpha	0.001,0.01,0.1	0.001
gamma	0.001,0.01,0.1	0.001
lambda	0.001,0.01,0.1	0.1
learning_rate	0.001,0.01,0.1	0.1
max_depth	3,6,12	12
n_estimators	50,100,200	200
subsample	0.5,0.7,1	0.5

3) Evaluation: The optimal XGB model achieved a train accuracy of 1.00 and a test accuracy of 0.81097, which indicates some level of overfitting, and still performs well on unseen data.

4.2.7 Histogram-Based Gradient Boosting (HGB)

1) Model Selection Rationale: Histogram-based Gradient Boosting (HGB) is a gradient-boosting algorithm using histogram-based techniques. Unlike traditional gradient boosting algorithms that find split points on sorted feature values, HGB buckets feature values into discrete bins and construct feature histograms [12]. This approach leads to reduced memory consumption and faster training times and is suitable for handling large datasets.

The efficiency of HGB in terms of memory consumption and training speed is particularly advantageous, especially when dealing with large datasets where computational resources are often a concern [13]. Additionally, the ability of HGB to achieve high predictive accuracy while maintaining faster training times makes it an attractive choice for this project.

2) Hyperparameter Optimization Results:

Hyperparameter	Pre-defined Values for Tuning	Best Values
max_depth	2,3,4,5,6,7,8,9,10	4
max_iter	40,80,160	80
learning_rate	0.001,0.01,0.1,0.3,0.5,0.7,0.9	0.1
max_leaf_nodes	3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25	21
l2_regularization	0.001,0.01,0.1	0.1

3) Evaluation: The optimal HGB model attained a training accuracy of 0.94 and a test accuracy of 0.84756, suggesting that the model performs well on unseen data.

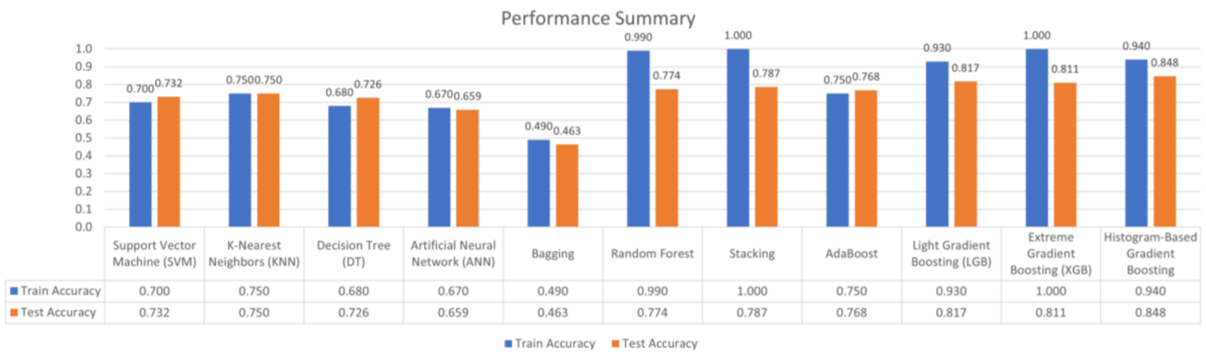
Out of all methods, HGB's test accuracy of 0.84756 is the highest, positioning it as the top-performing model in this dataset and securing the 101st position out of 1543 participants on the public score leaderboard in the Kaggle competition, positioned at top 6.5%. Screenshots of the Kaggle competition public score leaderboard result can be found in **Appendix A: Kaggle Competition Public Score Leaderboard Result.**

5. Performance Analysis

5.1 Performance Summary

The performance metrics for each method are summarized in Figure 10 below. Train Accuracy refers to the F1 micro-average score on the training set using 5-fold cross-validation. Test Accuracy is the public score on the Kaggle Competition leaderboard.

Figure 10: Performance Summary



	Support Vector Machine (SVM)	K-Nearest Neighbors (KNN)	Decision Tree (DT)	Artificial Neural Network (ANN)	Bagging	Random Forest	Stacking	AdaBoost	Light Gradient Boosting (LGB)	Extreme Gradient Boosting (XGB)	Histogram-Based Gradient Boosting
■ Train Accuracy	0.700	0.750	0.680	0.670	0.490	0.990	1.000	0.750	0.930	1.000	0.940
■ Test Accuracy	0.732	0.750	0.726	0.659	0.463	0.774	0.787	0.768	0.817	0.811	0.848

Comparing the performance of different methods reveals interesting insights:

1) Single Classifier Models: K-Nearest Neighbours (KNN) emerged as the top performer among single classifier models, achieving a test accuracy of 0.750. It was closely followed by a Support Vector Machine (SVM) with a test accuracy of 0.732. Decision Tree (DT) exhibited comparatively lower performance, with a train accuracy of 0.680 and a test accuracy below both KNN and SVM. Artificial Neural Network (ANN) displayed the lowest train and test accuracies, both falling below 0.68.

2) Ensemble Methods: Histogram-Based Gradient Boosting (HGB) showcased the highest test accuracy among ensemble methods, reaching 0.848. Light Gradient Boosting (LGB) and Extreme Gradient Boosting (XGB) followed closely behind, with test accuracies of 0.817 and 0.811, respectively. The top 3 performers are all based on gradient-boosting framework. Followed by Stacking, which also delivered promising results with a test accuracy of 0.787. Random Forest and AdaBoost exhibited slightly lower test accuracies of 0.774 and 0.768, respectively. However, Bagging displayed the lowest test accuracy of only 0.463, even performing worse than individual classifiers.

While single classifier models like KNN and SVM demonstrate decent accuracy, ensemble methods such as HGB, LGB and XGB outperform them, highlighting the importance of leveraging collective intelligence for the improved result of predictions. Among all methods, HGB emerges as the top performer with the best accuracy of 0.848.

5.2 Discussions

Performance metrics are crucial for achieving better outcomes. However, it's important to consider various factors such as overfitting and generalization, computational complexity and interpretability when determining the most appropriate model.

1) Overfitting and Generalization:

Ensemble methods excel in capturing intricate patterns and relationships within the data, leading to high predictive accuracy. This advantage is particularly beneficial for complex datasets where single classifier models may struggle to generalize effectively. However, certain ensemble methods such as Random Forest, Stacking, and XGB exhibit signs of overfitting, characterized by higher train accuracy but lower test accuracy.

2) Model Interpretability:

Single classifier models such as Decision Trees are inherently more interpretable due to their straightforward decision-making process. However, ensemble methods like Histogram-Based Gradient Boosting and Stacking, while providing better predictive accuracy, may sacrifice interpretability due to their ensemble nature. Model interpretability is essential in domains where stakeholders or domain experts need to understand the reasoning behind predictions.

3) Computational Complexity:

Ensemble methods tend to be more computationally intensive due to the training and combination of multiple models, which require significant computational resources but offer high performance. Conversely, single classifier models are less computationally demanding.

6. Conclusion

In this project, various single classifier models and ensemble methods were explored to address the classification problem of predicting horse health. Through experimentation and evaluation, valuable insights were gained into the strengths, weaknesses, and performance of each method.

Among the single classifier models, K-Nearest Neighbours (KNN) emerged as the top performer, closely followed by the Support Vector Machine (SVM). Decision Tree (DT) and Artificial Neural Network (ANN) displayed comparatively lower performance.

Ensemble methods outperformed single classifier models, with Histogram-Based Gradient Boosting (HGB) showcasing the highest test accuracy, followed closely by Light Gradient Boosting (LGB) and Extreme Gradient Boosting (XGB). Stacking also delivered promising results. Bagging exhibited the lowest test accuracy among all methods.

Overall, Histogram-Based Gradient Boosting (HGB), with a test accuracy of 0.84756, stands out as the most suitable model for this problem due to its superior accuracy performance, robustness on unseen data, and manageable computational cost. In this competition, where high accuracy is paramount, interpretability takes a back seat, making HGB's exceptional predictive power the key consideration. Moreover, HGB strikes a fine balance between performance and practicality, making it a preferred choice over other methods in this project.

References

[1] "F1 Score in Machine Learning," 27 Dec 2023. [Online]. Available: https://www.geeksforgeeks.org/f1-score-in-machine-learning/. [Accessed 23 Mar 2024].

[2] D. K, "Top 4 advantages and disadvantages of Support Vector Machine or SVM," 14 Jun 2019. [Online]. Available: https://dhirajkumarblog.medium.com/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107. [Accessed 30 Mar 2024].

[3] "What are the advantages and disadvantages of k-nearest neighbors for classification?," LinkedIn, [Online]. Available: https://www.linkedin.com/advice/1/what-advantages-disadvantages-k-nearest-neighbors. [Accessed 30 Mar 2024].

[4] "8 Key Advantages and Disadvantages of Decision Trees," Inside Learning Machines, [Online]. Available: https://insidelearningmachines.com/advantages_and_disadvantages_of_decision_trees/. [Accessed 30 Mar 2023].

[5] "Artificial Neural Networks Advantages and Disadvantages," *Mesopotamian Journal of Big Data 2021,* Vols. 29-31, 2018.

[6] R. Ragini, "Bagging and Boosting method," 6 Jun 2019. [Online]. Available: https://medium.com/@ruhi3929/bagging-and-boosting-method-c036236376eb. [Accessed 6 Apr 2024].

[7] "What are the Advantages and Disadvantages of Random Forest?," geeksforgeeks, 15 Feb 2024. [Online]. Available: https://www.geeksforgeeks.org/what-are-the-advantages-and-disadvantages-of-random-forest/. [Accessed 06 Apr 2024].

[8] Y. Lim, "Stacked Ensembles — Improving Model Performance on a Higher Level," 25 Mar 2022. [Online]. Available: https://towardsdatascience.com/stacked-ensembles-improving-model-performance-on-a-higher-level-99ffc4ea5523. [Accessed 6 Apr 2024].

[9] A. Dawari, "All About Adaboost," 3 May 2022. [Online]. Available: https://pub.towardsai.net/all-about-adaboost-ba232b5521e9. [Accessed 6 Apr 2024].

[10] E. Khandelwal, "Which algorithm takes the crown: Light GBM vs XGBOOST?," 22 May 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/. [Accessed 6 Apr 2024].

[11] S. Gupta, "Pros and cons of various Machine Learning algorithms," 28 Feb 2020. [Online]. Available: https://towardsdatascience.com/pros-and-cons-of-various-classification-ml-algorithms-3b5bfb3c87d6. [Accessed 13 Apr 2024].

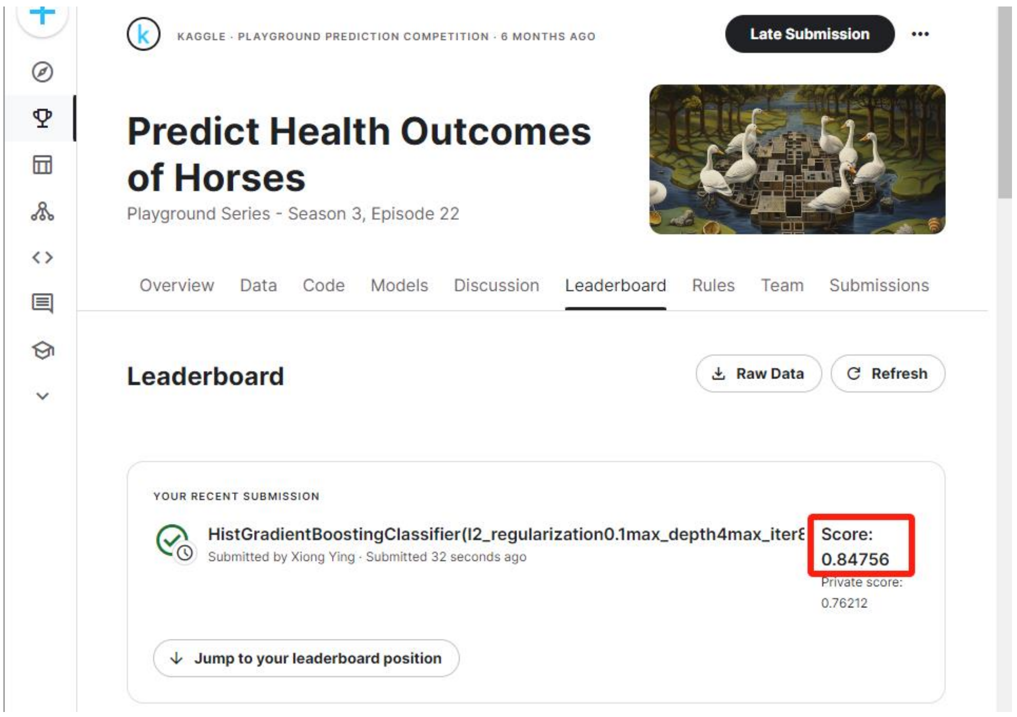
[12] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, Long Beach, California, USA, 2017.

[13] P. S, "Histogram Boosting Gradient Classifier," 15 Mar 2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2022/01/histogram-boosting-gradient-classifier/. [Accessed 13 Apr 2024].

[14] M. Y. H, "https://www.kaggle.com/datasets/yasserh/horse-survival-dataset," [Online]. Available: https://www.kaggle.com/datasets/yasserh/horse-survival-dataset. [Accessed 23 Mar 2024].

Appendix A: Kaggle Competition Public Score Leaderboard Result

Screenshot of Public Score on Kaggle Public Score Leaderboard:



KAGGLE · PLAYGROUND PREDICTION COMPETITION · 6 MONTHS AGO

Late Submission

Predict Health Outcomes of Horses

Playground Series - Season 3, Episode 22

Overview Data Code Models Discussion Leaderboard Rules Team Submissions

Leaderboard

Raw Data Refresh

YOUR RECENT SUBMISSION

HistGradientBoostingClassifier(l2_regularization0.1max_depth4max_iter8

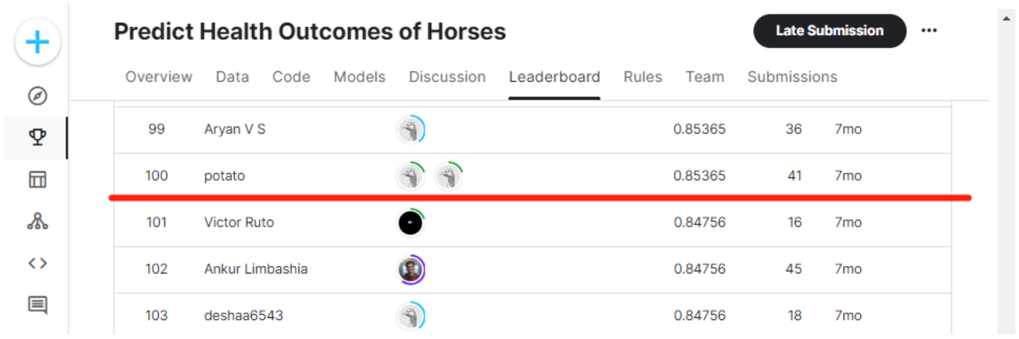
Submitted by Xiong Ying · Submitted 32 seconds ago

Score: 0.84756

Private score: 0.76212

Jump to your leaderboard position

Screenshot of Ranking on Kaggle Public Score Leaderboard:



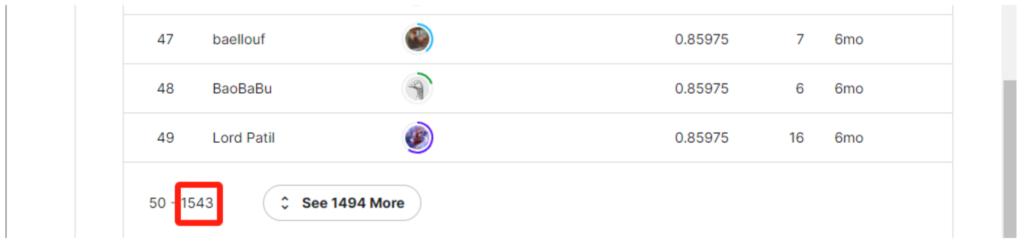
Predict Health Outcomes of Horses

Late Submission

Overview Data Code Models Discussion Leaderboard Rules Team Submissions

99	Aryan V S	0.85365	36	7mo
100	potato	0.85365	41	7mo
101	Victor Ruto	0.84756	16	7mo
102	Ankur Limbashia	0.84756	45	7mo
103	deshaa6543	0.84756	18	7mo

Screenshot of Total Number of Participants on Kaggle Public Score Leaderboard:



47	baellouf	0.85975	7	6mo
48	BaoBaBu	0.85975	6	6mo
49	Lord Patil	0.85975	16	6mo
50 - 1543	See 1494 More			