Statistical Learning & Pokemon

Victor Ma

5/11/2025

Introduction Statement

This analysis focuses on the "Pokemon" dataset, which contains information about the fictional creatures (these creatures fall under the umbrella of a 'Pokemon') from Nintendo's popular video game series. Each observation represents one Pokemon 'species', with various features that define their characteristics.

The primary question being asked is: How accurately can we predict a Pokemon's first type (type_1) based on numerical explanatory variables (height, weight, hp, attack, defense, speed) and categorical explanatory variables (color_1, egg_group_1, generation_id)?

Understanding the relationship between a Pokemon's attributes and its type has real-life applications for game development and provides insights into what goes into successful game design. This analysis could reveal patterns or strategies that game developers utilize to build an engaging video game that has users continuously coming back.

Data Cleaning & Wrangling

This project aims to predict the primary type (type_1) of a Pokemon based on its various characteristics. The "Pokemon" dataset that we are using contains information on numerous Pokemon species, including their combat statistics, physical attributes, and even color. Accurately predicting a Pokemon's primary type can offer insights into game design principles and the underlying patterns that go into a successful video game franchise.

For variables like "type_2", "color_2", "color_f", and "egg_group_2", where "NA" indicates the absence of the attribute rather than missing information, binary indicator variables ("has_type_2", "has_color_2", "has_color_f", and "has_egg_group_2") were created.

The "attack" and "special_attack" attributes were combined into "total_attack". Similarly, "defense" and "special_defense" were combined into "total_defense" to potentially reduce multicollinearity and capture overall offensive/defensive capabilities. Unique identifiers like "id" and "species_id" were removed to prevent overfitting.

The original response variable "type_1" has 18 levels. To simplify the classification task and focus on more prevalent types, the dataset was filtered to include only the four most common primary types. After some testing, it was concluded that "color_1" should be removed due to overfitting. It seems like Pokemon tend to carry a specific color that could easily identify certain species. "generation_id" and "type_1" were formally converted into factors, and observations with NA in "generation_id" were dropped.

The following table contains the variables ultimately selected for our models.

Table 1: Definitions of Variables

Name	Variable	Role	Definition	Values
height	Explanatory	Numerical	The height of each pokemon.	Number Value
weight	Explanatory	Numerical	The weight of each pokemon.	Number Value
base_experience	Explanatory	Numerical	The base experience of each Pokemon.	Number Value
type_1	Response	Categorical	The primary type.	1 out of 4 different types
has_type_2	Explanatory	Categorical	Whether it has a secondary type.	Yes/No (0 or 1)
hp	Explanatory	Numerical	The HP (hit points).	Number Value
total_attack	Explanatory	Numerical	The total attack points.	Number Value
total_defense	Explanatory	Numerical	The total defense points.	Number Value
speed	Explanatory	Numerical	The speed.	Number Value
has_color_2	Explanatory	Categorical	Whether it has a secondary color.	Yes/No (0 or 1)
has_color_f	Explanatory	Categorical	Whether it has a final color.	Yes/No (0 or 1)
egg_group_1	Explanatory	Categorical	The primary egg group.	1 out of 15 egg groups
has_egg_group_2	Explanatory	Categorical	Whether it has a secondary egg group.	Yes/No (0 or 1)
generation_id	Explanatory	Categorical	The generation ID of each Pokemon.	Value 1-7 Indicating Generation

Exploratory Data Analysis

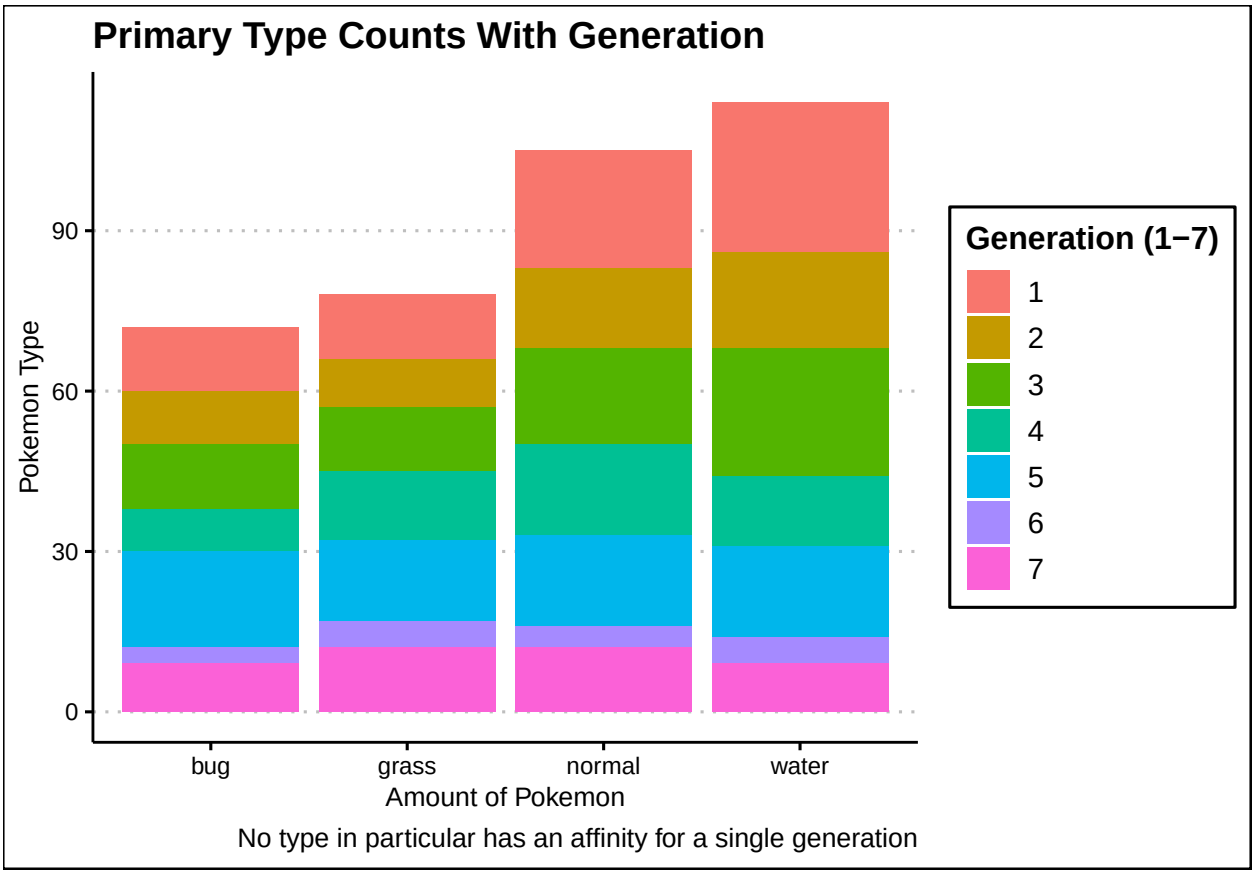

Figure 1: Distribution of Primary Types across generations. This bar chart showcases the count of Pokemon for each of the four most common primary types across different generations. It indicates that no particular type shows a strong affinity for a specific generation, suggesting a consistent distribution of these types over time.

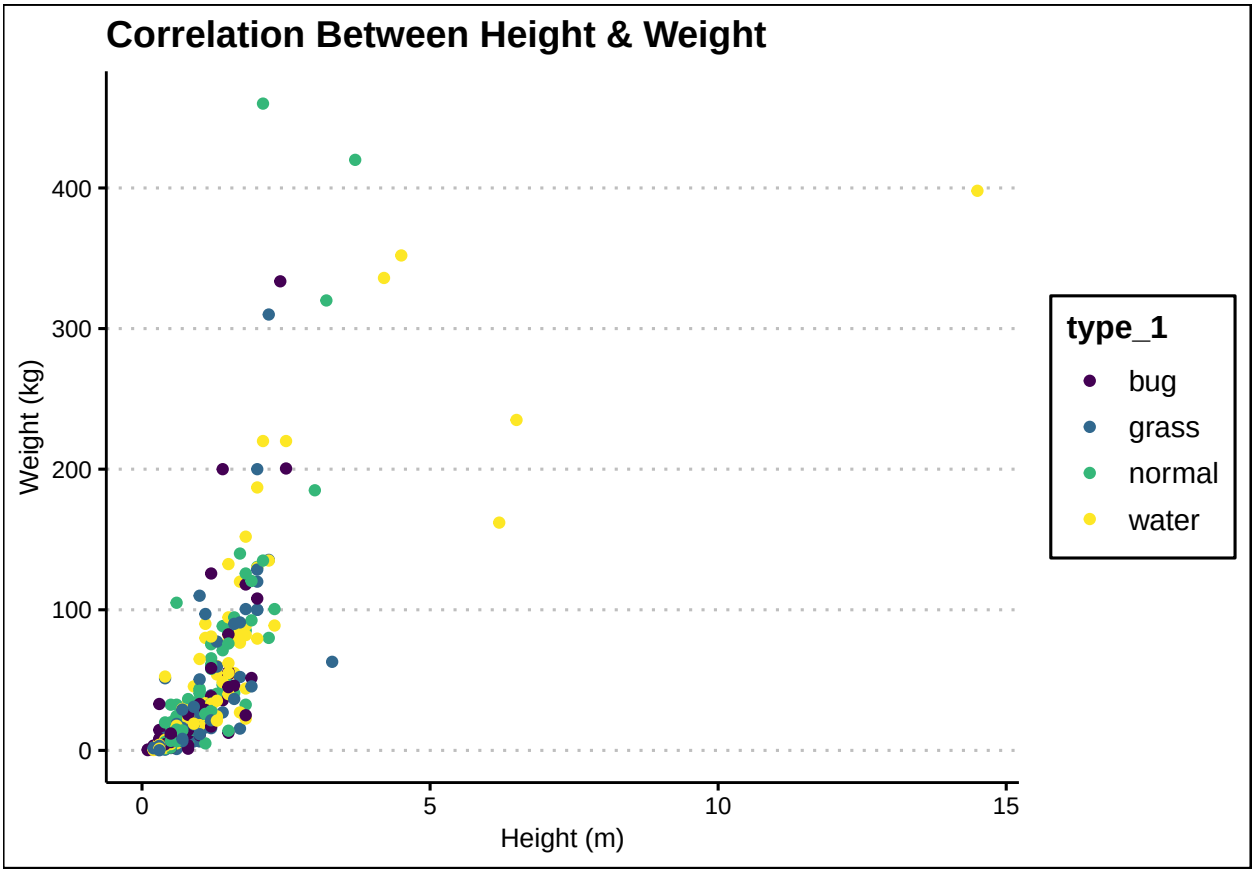

Figure 2: Relationship between Height and Weight by Primary Type. This scatter plot shows the correlation between Pokemon height (meters) and weight (kilograms), with points colored according to their primary type. The plot shows a generally positive correlation between height and weight, and also shows how different Pokemon types are distributed across the board.

Model Building

When trying to predict a Pokemon's primary type based on multiple features, I used Multinomial LASSO Regression and Random Forest Classification. Each model was optimized using cross-validation.

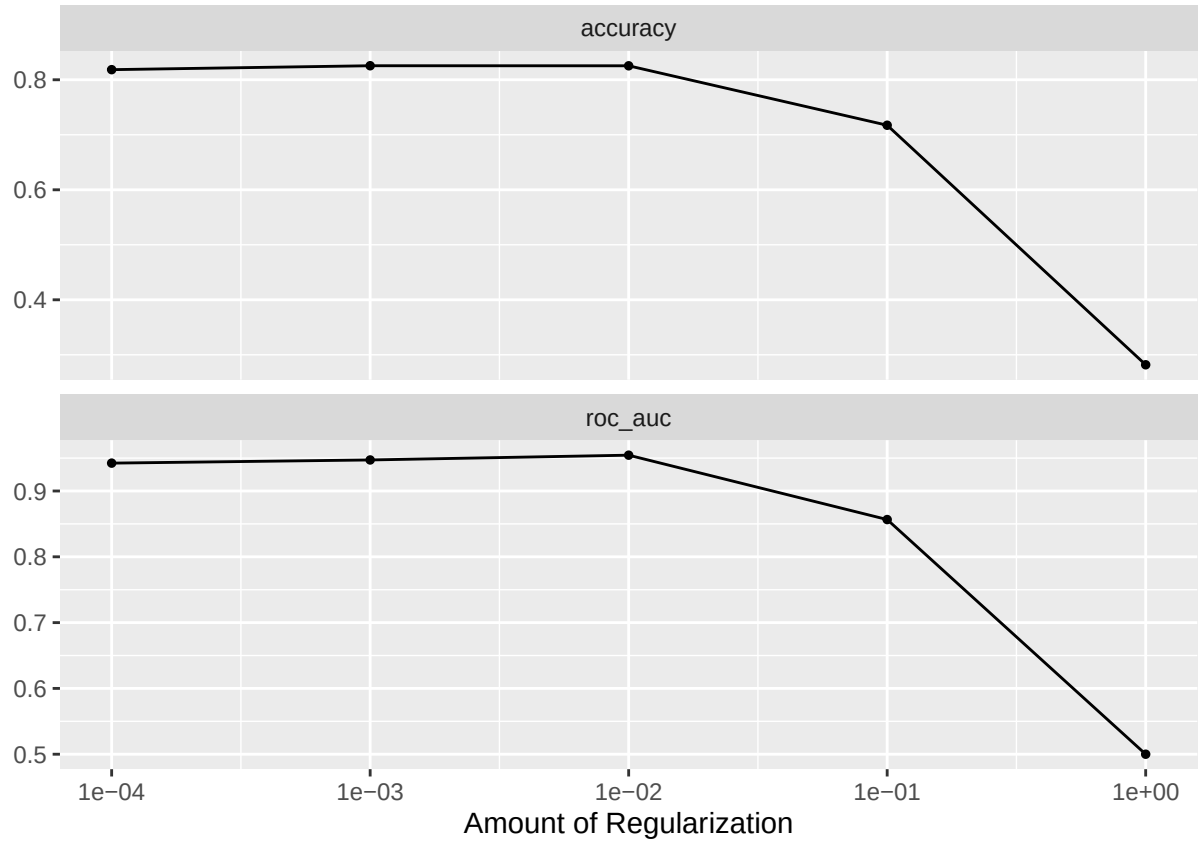

Figure 3: The top panel shows accuracy, and the bottom panel shows ROC AUC. Optimal performance appears to be achieved at around 1e-02 regularization, after which performance declines as regularization increases too much.

```
## # A tibble: 6 x 7
##    mtry .metric  .estimator  mean     n std_err .config
##   <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1     2 accuracy multiclass 0.786    10  0.0249 Preprocessor1_Model1
## 2     2 roc_auc  hand_till  0.923    10  0.0151 Preprocessor1_Model1
## 3     7 accuracy multiclass 0.815    10  0.0193 Preprocessor1_Model2
## 4     7 roc_auc  hand_till  0.943    10  0.0104 Preprocessor1_Model2
## 5    13 accuracy multiclass 0.815    10  0.0210 Preprocessor1_Model3
## 6    13 roc_auc  hand_till  0.950    10  0.0104 Preprocessor1_Model3
```

```
## # A tibble: 3 x 7
##    mtry .metric .estimator  mean     n std_err .config
##   <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1    13 roc_auc hand_till  0.950    10  0.0104 Preprocessor1_Model3
## 2     7 roc_auc hand_till  0.943    10  0.0104 Preprocessor1_Model2
## 3     2 roc_auc hand_till  0.923    10  0.0151 Preprocessor1_Model1
```

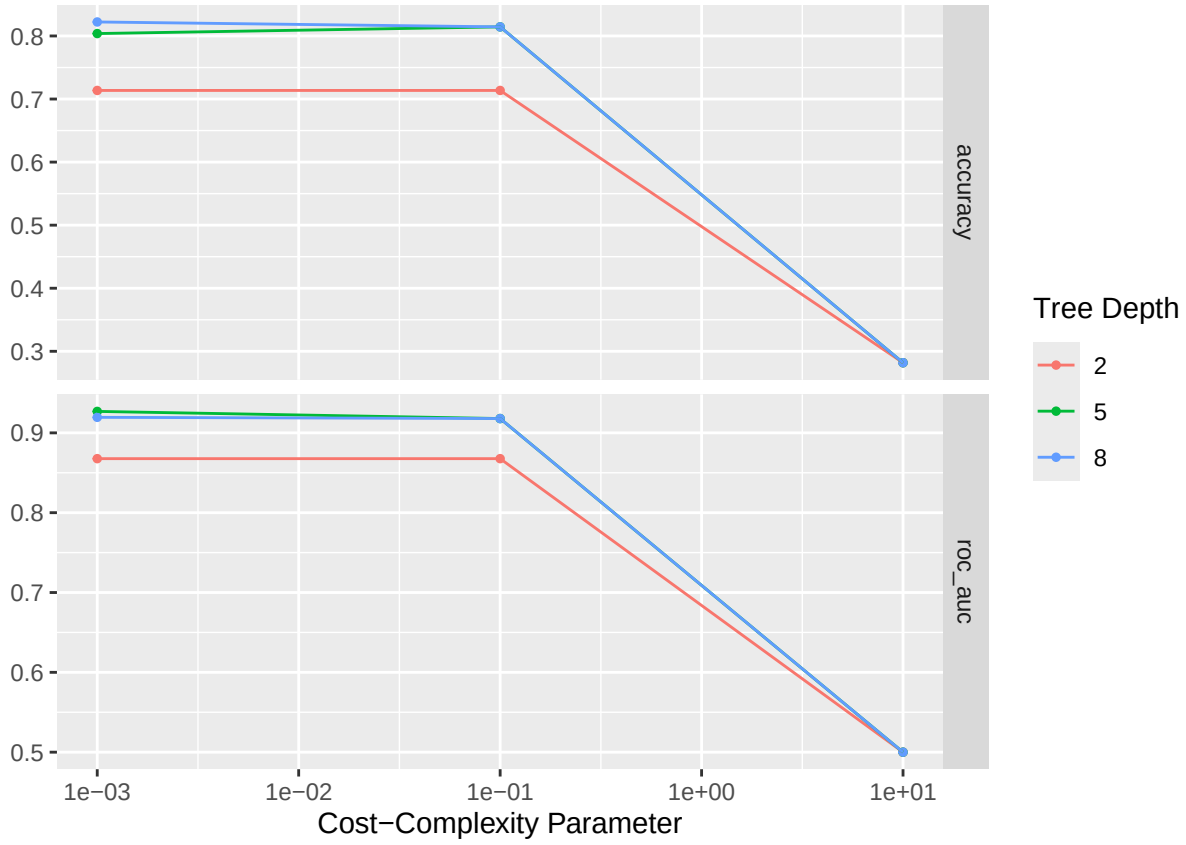

Figure 4: This graph shows the impact of the cost-complexity parameter on the accuracy and ROC AUC of our Decision Tree model. The plots show that higher cost-complexity values lead to simpler trees and lower performance after a certain point, here around 1e-01.

```
## # A tibble: 6 x 7
##    mtry .metric  .estimator  mean     n std_err .config
##   <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1     2 accuracy multiclass 0.786    10  0.0249 Preprocessor1_Model1
## 2     2 roc_auc  hand_till  0.923    10  0.0151 Preprocessor1_Model1
## 3     7 accuracy multiclass 0.815    10  0.0193 Preprocessor1_Model2
## 4     7 roc_auc  hand_till  0.943    10  0.0104 Preprocessor1_Model2
## 5    13 accuracy multiclass 0.815    10  0.0210 Preprocessor1_Model3
## 6    13 roc_auc  hand_till  0.950    10  0.0104 Preprocessor1_Model3
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.785
```

```
##            Truth
## Prediction bug grass normal water
##     bug     12     0      1     0
##     grass    1    15      4     2
##     normal   0     5     20     0
##     water    2     3      2    26
```

Model Refinement

The three models were trained on the training dataset "pokemon_train_tbl", and hyperparameters were optimized using 10-fold cross-validation, selecting the best ROC_AUC value, and using the 'One Standard Error' rule. The final models, using our optimized hyperparameters, were then evaluated on the test set "pokemon_test_tbl" to get a final measure of their performance.

My models seem to fit quite well, but only after much data wrangling and manipulation of the original variables. There could have been many are many different approaches taken, but the models we have as of now showcase that physical attributes and combat statistics allow us to quite accurately predict the primary type of a Pokemon species. Our initial variable selection and manipulation such as combining attack and defense statistics, as well as removing "color_1" was crucial for the performance of our models.

After tuning, the LASSO model was finalized with a penalty of 0.01. The Decision Tree model was finalized with a tree_depth of 5 and a cost_complexity of 0.1. The Random Forest model was finalized with an mtry value of 13.

Conclusion

Based on the test set accuracy, the Multinomial LASSO Regression model performed the best, with an accuracy of 81.5%, precision of 83.7%, and a sensitivity of 80.9%. According to the confusion matrix, our model had the least amount of trouble predicting 'bug' types, while having a bit of trouble with 'grass' and 'normal' types. Due to the nature of the video game, it is possible that 'bug' Pokemon tend to share more similar traits in comparison to other Pokemon types among each other (small, green, weak, etc.). The most important variables in our best model were egg_group_1, generation_id, hp, has_egg_group2, and total_attack.

This project successfully explored the predictability of a Pokemon's primary type (type_1) using various attributes such as combat statistics, physical characteristics, and categorical identifiers. Through comprehensive data cleaning, wrangling, and application of the machine learning models we learned from class, we were able to uncover patterns beneficial for successful game development.

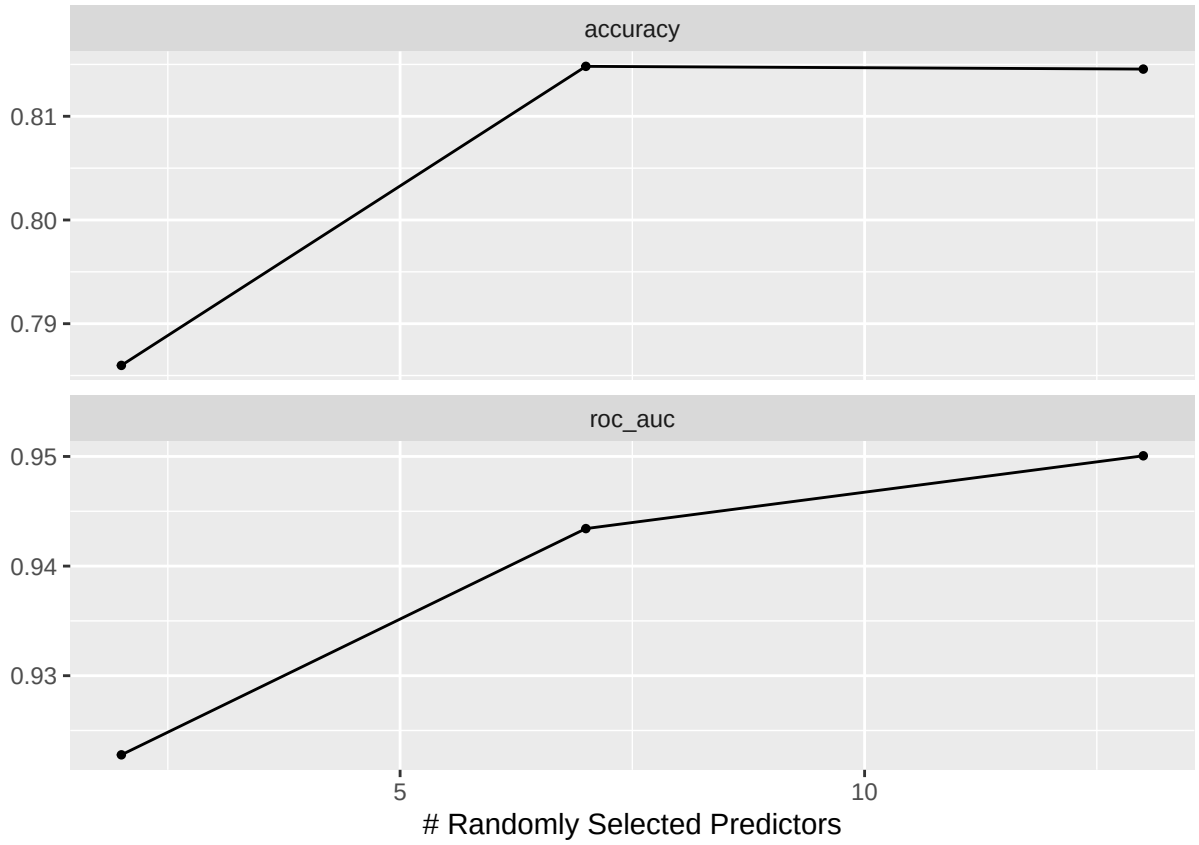

Figure 5: This graph shows how the number of randomly selected predictors (mtry) influences the accuracy and ROC AUC of our Random Forest model. Optimal performance appears to be achieved at around the value of 13.

```
##           Truth
## Prediction bug grass normal water
##     bug     12     0      0     0
##     grass    1    16      3     2
##     normal   0     5     22     1
##     water    2     2      1    25
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.815
```

```
## # A tibble: 1 x 3
##   .metric   .estimator .estimate
##   <chr>     <chr>          <dbl>
## 1 precision macro          0.837
```

```
## # A tibble: 1 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 sensitivity macro          0.809
```

Table 2: Model Comparison Summary

Model	Accuracy	Most.Important.Variables
Multinomial LASSO Regression	0.815	egg_group_1, generation_id, hp, has_egg_group2, total_attack
Classification Decision Tree	0.753	egg_group_1, total_attack, has_egg_group_2, height, speed
Random Forest Classification	0.774	egg_group_1, has_egg_group_2, hp, generation_id, weight