# 4 Difficulty Predictor

The method for making the difficulty predictor is based on the work of Kreveld et al. [16] but with some modifications. The goal is to have a predictor that, given a level, can estimate how difficult it is. To implement this, Kreveld et al. used linear regression and focused on selecting a few relevant variables. We built upon this by using other linear regression tools and iteratively selecting what variables to use.

## 4.1 Tools and method

The script for training was written in Python using the scikit-learn library for the machine learning methods. From this basic linear regression, lasso regression and ridge regression were used. The Pandas library was used for data management. The data consisted of the 250 levels from Longcat with variables derived from their graph representation and the levels' average completion time. There were some deviations between the iterations, but generally, the data was split into 80% training data and 20% test data. Each training was run 50 times with new randomizations of the split, and the Mean Aquared error (MSE) results were averaged over all runs.

## 4.2 Predictor Development

In this section we describe the development process of the predictor. We go through how each iteration was developed, their problems and how the next iteration was created to solve those problems. A summary of each iteration what method was used, what variables were used, and what MSE it gave can be seen in Figure 4.1.

### Iteration 1: Simple Linear Regression

The first iteration was very close to the method presented by Kreveld et al. [16], but without selection. We used linear regression with the first seven variables from the analysis. The resulting model had a MSE of 204.73. Only the first seven variables where used since the others where not discovered yet.

### Iteration 2: Simple Linear Regression with variable selection

For the second iteration, we followed the instructions of Kreveld et al. more closely and removed variables that we believed were less relevant. We removed *Size* since it seemed to have a weak correlation. Also, we removed *Number of Fails*. The intuition for *Number of Fails* was that more fails would give a more difficult level, but in practice, it is significantly correlated with *Number of states*, giving very little additional prediction power. The resulting model had an MSE of 199.00. A slight improvement from the previous.

### Iteration 3: Lasso Regression

The third iteration used the same variables as iteration 2, but used Lasso regression. To find a good alpha, cross-validation was used. For the cross-validation, the training data was split into 5 random sets, and 1000 was used as the maximum iteration. The resulting model had an MSE of 200.75.

### Iteration 4: Ridge Regression

The fourth iteration used the same variables as iterations 2 and 3, but used ridge regression. The same method for finding an alpha value was used as in iteration 3. The resulting model had an MSE of 181.89. This was a substantial improvement, so we continued iterating with ridge regression.

At this stage of the implementation, further analysis was conducted to understand what types of levels the model fails at. During this analysis, we found that some levels had a large number of states but a short average completion time. We played the levels and found that they were easy, as it was clear that certain actions led to a state where a solution was impossible, i.e. dead states. To exploit this understanding, we created the concept of dead states and indeterminate states. Indeterminate states came as a consequence of dead states. Since dead states are states that don't increase difficulty, we really want to measure the number of non-dead states, but this would also include solution states, and a level of only solutions is trivial. Hence, the concept of indeterminate states was created, which is non-dead and non-solution states.

### Iteration 5: Indeterminate States

The fifth iteration used the same set-up as iteration 4, but used the variable "Number of indeterminate" states instead of "Number of states." We also found that both the ridge regression and lasso regression minimized "Number of Branches", so it was also removed for this iteration. The resulting model had an MSE of 150.76. This showed a significant improvement, so it could be assumed that the analysis of dead states and indeterminate states is correct.

At this stage, we started work with the generators, using the model from iteration 5. Initially, it was used to categorize levels created by the random generator, and later to train the reinforcement learning methods. However, when playtesting levels, we found unwanted patterns. Some generated levels had many indeterminate states, but were still easy. These levels had "tunnels" that created a long section of indeterminate states where the player could only make one choice. The length of these sections didn't add complexity, but added more indeterminate states, which the predictor evaluated as making the level more difficult. An example of one of these "tunnel" levels can be seen in Figure 4.1. We can see the tunnel with many turns on the left side of the level, and the resulting additional linear indeterminate states in the graph representation. This problem was not present within the original non-generated levels, as the levels were selected not to have tunnels. Most likely, since tunnels just make the level longer but not more fun. To avoid this problem, we chose to use indeterminate branches instead. This solves the problem since branches count states with a out degree larger than one.

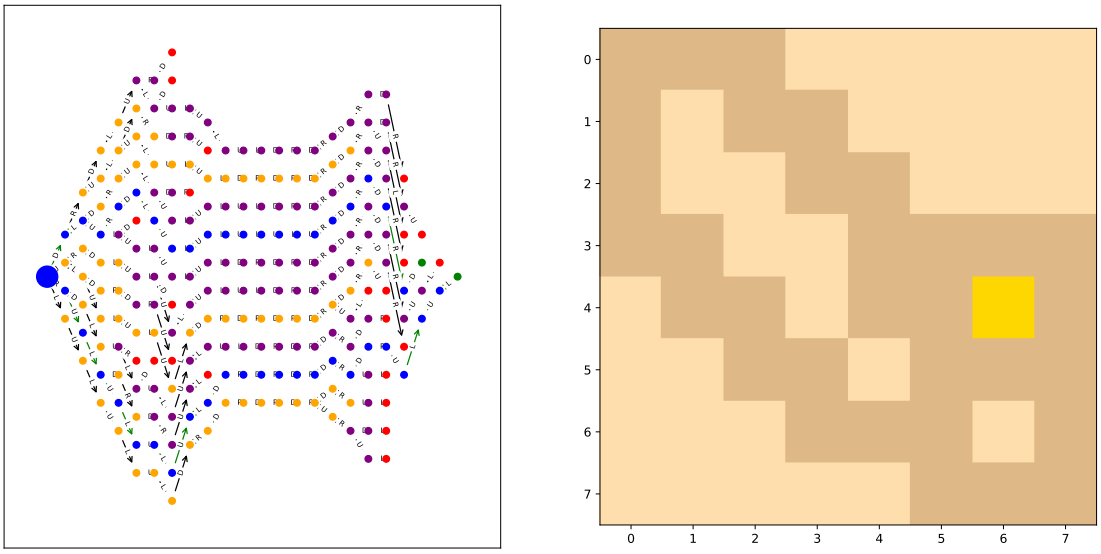


Figure 4.1: Generated level with a long tunnel.

So the tunnels of linear indeterminate states wont add to indeterminate branches while still having the same strengths as indeterminate states.

### Iteration 6: Indeterminate Branches

The sixth iteration used the same set-up as iteration 5 but used the variable "Number of indeterminate branches" instead of "Number of indeterminate". The resulting model had an MSE of 166.41. This is worse than the previous model, but the levels generated using this model were perceived to be more difficult when play-tested by us. The play-testing was very unstructured, with us playing levels rated as difficult and seeing if we found them in general more difficult than lower-rated levels. Using this predictor also resulted in fewer tunnels, so it was preferred over the the previous iteration.

Further work with the generators was done, and even though the tunnel problem was reduced, it still persisted. We found that since the variable "Shortest solution length" rewarded the generator for these tunnels, as each state in the tunnel made the solution longer, but it didn't add to the difficulty. We can also see this in the Figure 4.1 as the solutions there have long stretches of solution states, in blue, that would be the same for the player as if there were fewer. The generators could also generate very simple levels that consisted only of a tunnel, but predicted them to be somewhat difficult. To mitigate this, we removed the variable.

### Iteration 7: Removing Shortest Solution Length

The seventh iteration used the same set-up as iteration 6 but didn't use the variable "Shortest solution length". The resulting model had an MSE of 187.76. This is also worse than the previous model, but we found that the generated levels were now much more accurately evaluated, as previously over-evaluated tunnel levels were now more accurately categorized as trivial.

| Iter. | Method | N States | Size | N Bran. | N sol. | N Fails | Sol. len | N sol. bran. | N ind. | Ind bran. | MSE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Linear | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | 204.73 |
| 2 | Linear | ✓ | ✗ | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ | 199.00 |
| 3 | Lasso | ✓ | ✗ | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ | 200.75 |
| 4 | Ridge | ✓ | ✗ | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ | 181.89 |
| 5 | Ridge | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ | 150.76 |
| 6 | Ridge | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✓ | ✗ | ✓ | 166.41 |
| 7 | Ridge | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ | 187.76 |

Table 4.1: Table showing each iteration of the difficulty predictor, what method was used, what variables were used, and what result it gave.