Difficulty Grading of a Kilter Board Route

1st Bonifác Lukács Faucher-Timár
Master's in Automatic Control and Robotics
Universitat Politecnia De Catalunya
Barcelona, Spain
bonifac.lukacs.faucher-timar@estudiantat.upc.edu

2nd Kavy Nihar Mittal
Master's in Automatic Control and Robotics
Universitat Politecnia De Catalunya
Barcelona, Spain
kavy.nihar@estudiantat.upc.edu

3rd Oriol Martínez Fité
Master's in Automatic Control and Robotics
Universitat Politecnia De Catalunya
Barcelona, Spain
oriol.martinez.fite@estudiantat.upc.edu

***Abstract*—In this project, we develop a machine learning model to predict the difficulty rating of bouldering problems on a Kilter Board. The Kilter Board is a standardized climbing wall connected to the app that enables climbers to create, share, and attempt problems with consistent hold layouts. Each problem is typically rated between 1 and 39, but these ratings are user-submitted and subjective, often leading to inconsistent difficulty assessments. Our system leverages route metadata—hold positions, wall angle—to estimate the likely difficulty level of a route. By automating difficulty estimation, this project aims to improve grading consistency, assist route setters, and enhance the experience of climbers by providing more reliable benchmarks for training and progression.**

***Index Terms*—**

I. Motivation and Importance

The popularity of indoor climbing has grown rapidly and, along with it, the use of training tools such as the Kilter Board. One of the board's strengths is its large library of user-created problems, which can be accessed and climbed anywhere in the world on identical hardware. However, the grading of these problems is crowd-sourced, subjective, and often inconsistent. This makes it difficult for climbers to accurately track progress or choose the appropriate challenges.

An objective data-driven method for estimating the difficulty of a Kilter Board problem can improve the reliability of the experience, helping climbers choose problems aligned with their ability and training goals. It can also help gym managers and trainers in curating better collections of problems.

II. Basic Concepts

A. What is a Kilter Board?

A Kilter Board, as can be seen in Fig. 1, is a digitally interactive climbing wall with a fixed grid of handholds and footholds, each backlit by LEDs. Climbers use an app to select routes, which light up only the holds used for a specific problem. The wall's angle is adjustable, allowing routes to be attempted at various inclines, from vertical to steeply overhung.

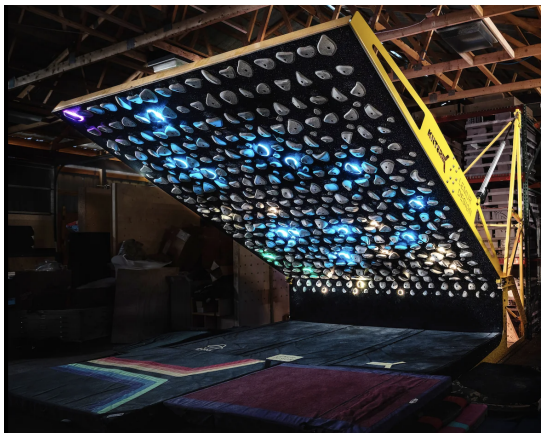

Fig. 1. Kilter Board

B. What is Difficulty Rating?

The difficulty of a climb is commonly rated using the V-grade scale, ranging from V0 (easiest) to V15+ (elite level). This scale attempts to capture the physical and technical challenges of a boulder problem, including the complexity of movements, size and shape of holds, and required strength or technique. Here we are using a rating between 1 and 39, which can be converted to V-grade using a conversion scale. However, due to the subjective nature of climbing and personal variability in skill, the same problem might receive different grades from different climbers.

Because the grading system is based on human perception, the ratings in our dataset are inherently subjective and inconsistent. As shown in Fig. 2, most ratings are given as whole numbers, with only a small fraction including decimal values. This lack of precision introduces variability and potential inaccuracies in the difficulty labels.

However, this subjectivity is precisely what our project aims to address. By analyzing a large volume of human-assigned ratings, we seek to develop a more standardized and objective grading model. Our approach leverages the collective input

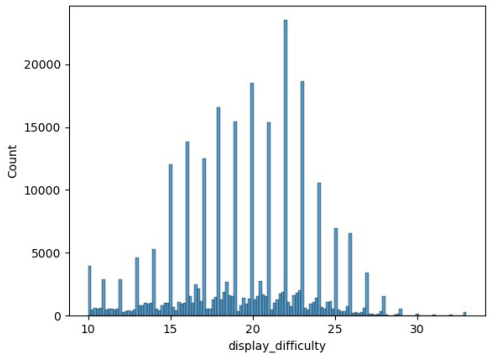

Fig. 2. Histogram of difficulty ratings for all climbs

of many climbers to account for individual biases, while preserving the richness of human judgment at the core of climbing difficulty assessments.

C. Factors Influencing Difficulty

There are several key features that influence the perceived and actual difficulty of a Kilter Board route:

- **Wall angle (tilt)**: Steep angles increase difficulty.
- **Hold size and type**: Smaller holds are harder to grip.
- **Hold position and spacing**: Longer reaches or awkward body positions increase the challenge.
- **Number of moves**: More moves generally require greater endurance or sequencing.
- **Foot placement constraints**: Problems that restrict foot use or require precise footwork may be more difficult.

These features are derived from the route's data (hold coordinates, wall angle, etc.) and analyzed using spatial and geometric techniques before being input to the model.

III. Dataset Description

A. Data Collection Method

To train and evaluate the machine learning model for predicting the difficulty of Kilter Board routes, we collected a dataset from the Kilter Board mobile application. Using an API, we extracted a SQL database which was then stored in .csv files.

B. Preprocessing and Feature Engineering

To prepare the raw dataset for machine learning, we performed several preprocessing steps and extracted features to capture the spatial and structural characteristics of each Kilter Board route.

1) Feature Construction: From the raw hold positions and wall angle, we derived a variety of geometric and structural features, including:

- The x coordinate of the hold position
- The y coordinate of the hold position
- If the hold is a start hold
- If the hold is a finish hold
- If the hold is a foothold
- Wall angle

Each hold is described using five features, and we also include one additional feature representing the angle of the climb. Therefore, the total number of features is calculated as:

$$\text{Total Features} = (\text{Number of Holds} \times 5) + 1 \tag{1}$$

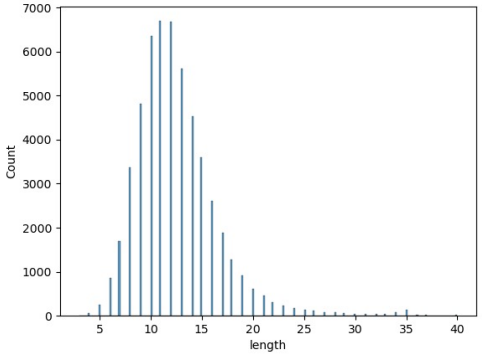

Fig. 3. Histogram of the total holds for all climbs

As shown in Fig. 3, climbs with more than 20 holds are relatively uncommon—only about 2,000 out of the full dataset. Based on this observation, we limit the maximum number of holds to 20. This cap reduces sparsity in our feature space and allows for more consistent model performance.

To ensure reliable grading, we also filter for climbs that have been completed at least five times. This helps us exclude problems with insufficient user feedback, which might result in unreliable or overly subjective difficulty ratings.

After applying these criteria, our final dataset contains approximately 50,000 climbs. With the 20-hold cap, the number of features per climb becomes:

$$(20 \times 5) + 1 = 101 \tag{2}$$

This is a reduction from the 201 features that would have resulted from allowing up to 40 holds.

The Fig. 4, shows the distribution of the number of climbs for different angles of the Kilter board, where it can be observed that climbs have been attempted at all angles, with the most preferred angles falling between 15 and 50 degrees.

2) Vector Representation: Each climbing problem is represented as a high-dimensional vector that encodes the spatial and structural properties of the route. After capping the maximum number of holds at 20, each problem is encoded as a vector of **101 dimensions**: 20 holds, each with five descriptive features, plus one additional feature representing the overall angle of the wall. This representation captures both individual hold attributes and broader route-level patterns, allowing the model to generalize across different climb configurations.

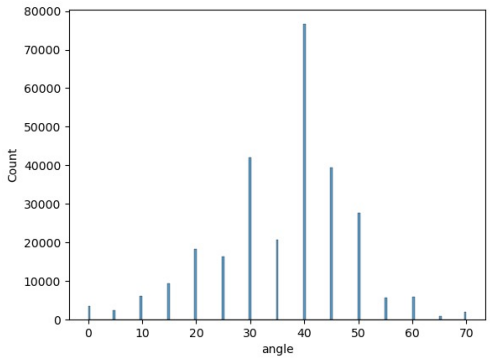

Fig. 4. Histogram of angles for all clmibs

3) Scaling: The 101 dimensional feature representing each climb was then scaled, so that the features are within some bounds that the models can handle easier. For the 100 hold features standard scaling was used, so that the resulting distributions are standard. For the wall angle and the difficulty minmax scalers were used to have these values in the (0, 1) range. This is a common step in machine learning and it helps model performance.

4) Dimensionality Reduction Using PCA: To reduce feature redundancy and enhance computational efficiency, we applied **Principal Component Analysis (PCA)** to the hold-level features. PCA is a linear transformation technique that identifies the directions (principal components) along which the variance in the data is maximized. By projecting the original high-dimensional data onto these components, PCA captures the most informative aspects of the dataset while discarding noise and less relevant variation.

In our case, the original 5-dimensional feature vector for each hold was reduced to 2 principal components. This transformation decreased the overall feature space from 101 to **41 dimensions**, while preserving over **99.99% of the total variance** in the data.

The resulting lower-dimensional representation retained nearly all meaningful information, while significantly reducing computational cost and helping to prevent overfitting—particularly important when working with high-dimensional inputs.

IV. Training and Testing Data

For training and testing our machine learning models, we split the dataset into training and testing sets. The training set consists of input features (X_{train}) and corresponding difficulty ratings (Y_{train}), while the testing set consists of input features (X_{test}) and corresponding difficulty ratings (Y_{test}) used for model evaluation.

- **Training Data (X_{train}, Y_{train}):** The training set contains 43,164 samples ($X_{\text{train}} : 43164 \times 41$) and their corresponding difficulty ratings ($Y_{\text{train}} : 43164$). Each sample in X_{train} is represented by a 41-dimensional feature vector, capturing the geometric and structural properties of the Kilter Board route holds and the angle.
- **Testing Data (X_{test}, Y_{test}):** The testing set consists of 10,792 samples ($X_{\text{test}} : 10792 \times 41$) with their corresponding difficulty ratings ($Y_{\text{test}} : 10792$). Similar to the training data, each sample in X_{test} is represented by a 41-dimensional feature vector.

This dataset split ensures that the models are trained on a large portion of the data (around 80% of the total dataset) and evaluated on a separate portion (about 20%), providing an unbiased estimate of their performance on unseen data. The training data is used to teach the model the relationships between the features and the difficulty ratings, while the testing data is used to evaluate how well the model generalizes to new, unseen climbs.

V. Approach to Difficulty Rating Prediction

To predict the difficulty of Kilter Board routes, we employed various regression models and evaluated their performance using the **R^2 score**. The R^2 score indicates how well the model's predictions align with the actual difficulty ratings. An R^2 score of 1 means perfect predictions, while a score of 0 suggests no correlation.

We used the following regression models, each of which has specific parameters that affect its performance and predictive capabilities. The models tested include Linear Regressor, Decision Tree, Random Forest, Extra Tree, Gradient Boosting, and Hist Gradient Boosting.

Grid search was used to find the optimal parameters in the presented regression models because it systematically evaluates a range of hyperparameter combinations through cross-validation. This method ensures that the selected parameters generalize well across different subsets of the data, improving model performance and reducing the risk of overfitting. The parameters values for each model shown below are the ones found by grid search.

Below is an explanation of the key parameters used for each model:

A. Linear Regressor

Parameters: None.

- Linear regression assumes a linear relationship between the features and the target (difficulty rating). This model is simple and interpretable but may struggle to capture the non-linear complexities inherent in the climbing difficulty dataset.

B. Decision Trees

Parameters:

- `max_depth=10`: This parameter limits the depth of the decision tree to 10 levels. By controlling the depth, we prevent the tree from growing too complex and overfitting the training data. A deeper tree may learn very specific patterns in the data, but it may not generalize well to new, unseen data.

Decision Trees are non-linear models that split the feature space into regions based on the values of the input features. They are flexible and can model complex relationships, but they can overfit if the tree is too deep. In our case, the `max_depth` parameter was used to strike a balance between complexity and generalization.

C. Random Forest

Parameters:

- `max_depth=10`: Like in the Decision Tree Regressor, this parameter limits the depth of the trees in the random forest, which helps prevent overfitting.
- `n_estimators=1000`: This parameter defines the number of trees in the forest. A larger number of trees generally leads to better model performance, as the final prediction is averaged over many different decision trees. However, this comes with increased computational cost.

Random Forest is an ensemble method that combines multiple decision trees to improve accuracy and robustness. The `n_estimators` parameter determines how many trees are in the forest, while `max_depth` controls the depth of each individual tree. More trees typically lead to better performance, but the trade-off is higher computational cost.

D. Extra Trees

Parameters:

- `max_depth=10`: This limits the depth of the individual trees, helping to reduce the risk of overfitting.
- `min_samples_leaf=1`: This parameter controls the minimum number of samples required to be at a leaf node. Lower values allow the tree to be more complex, while higher values force the tree to be more generalized. Setting it to 1 means that every leaf can contain only one sample, allowing the tree to grow deeper.

Extra Trees (Extremely Randomized Trees) are similar to Random Forests but differ in how the trees are built. Instead of selecting the best split at each node, Extra Trees choose splits randomly. This randomness helps in reducing variance and overfitting, making the model more robust.

E. Gradient Boosting

Parameters:

- `learning_rate=0.1`: The learning rate controls how much each tree contributes to the final model. A lower learning rate makes the model slower to learn but can improve its precision and prevent overfitting. The value of 0.1 is a commonly used starting point.
- `n_estimators=1000`: This defines the number of trees in the boosting process. More trees generally lead to better model performance, but at the cost of increased computation time.

Gradient Boosting is an ensemble method that builds trees sequentially, where each tree corrects the errors made by the previous one. This method is effective for capturing complex relationships in the data. The `learning_rate` controls the contribution of each new tree, while `n_estimators` controls the number of trees used in the boosting process.

F. Histogram-based Gradient Boosting

Parameters:

- `learning_rate=0.1`: Like in the Gradient Boosting method, this controls how much each tree contributes to the final prediction. A learning rate of 0.1 allows the model to make progress without overfitting too quickly.
- `max_depth=8`: This limits the depth of the trees in the boosting process, striking a balance between complexity and generalization.
- `max_iter=1000`: This parameter defines the number of iterations (or boosting rounds). A higher number of iterations allows the model to fit more trees, which can lead to better predictions, but at the cost of increased computational time.

Hist Gradient Boosting is a more efficient version of Gradient Boosting, optimized for large datasets and high-dimensional data. It uses histogram-based methods to speed up training without sacrificing performance. This method is particularly effective for our problem, as it can handle the large number of features and data points efficiently.

Below are the **R^2 scores** for each of the models:

TABLE I
MODEL PERFORMANCE WITH R^2 SCORES

Model	R^2 Score
Linear Regressor	0.348
Decision Tree Regressor	0.499
Random Forest Regressor	0.593
Extra Tree Regressor	0.517
Gradient Boosting Regressor	0.810
Hist Gradient Boosting Regressor	0.860

- **Linear Regressor**: R^2 score of 0.348, which shows improvement over the baseline, but it is still a poor fit for the data.
- **Decision Tree Regressor**: R^2 score of 0.499, suggesting a better fit as it captures some non-linearities.
- **Random Forest Regressor**: R^2 score of 0.593, showing a noticeable improvement over Decision Tree.
- **Extra Tree Regressor**: R^2 score of 0.517, which is slightly better than the Decision Tree but less effective than Random Forest.
- **Gradient Boosting Regressor**: R^2 score of 0.810, showing a significant improvement, capturing complex patterns in the data.
- **Histogram-based Gradient Boosting Regressor**: R^2 score of 0.860, the best-performing model, demonstrating its ability to capture complex relationships while being computationally efficient.

VI. Neural network approach

A neural network approach was also tested on the training and testing data using the "Adam" optimizer, as it is very robust to changes in the architecture of the neural network. First a dummy neural network was used to tune the batch size for this dataset, which resulted in an optimal batch size of 3000. After that a general fully connected neural network architecture with dense and dropout layers was parameterized, then the hyperparameter space was searched using random search to find the models with the best performance. Random search was chosen over grid search in this case as the hyperparameter space has high dimensions. The resulting two best models were:

TABLE II
Neural Network Performance with R^2 Scores

$\mathbf{R}^2$ Score	Layers
0.631	$[360, 328, 480, 384, 88, 208, 1]$
0.621	$[216, 432, 232, 152, 336, 1]$
0.621	$[480, 440, 320, DROPOUT, 1]$

These models didn't perform as well as the models in the previous section therefore it was not examined further what architectures result in better estimations.

VII. Conclusion

After evaluating multiple regression models, the **Histogram-based Gradient Boosting Regressor** demonstrated the best performance, achieving an R^2 score of 0.860. This outperformed all other models, including Gradient Boosting and Random Forest, which had R^2 scores of 0.810 and 0.593, respectively. It's ability to efficiently capture complex relationships in the data, while maintaining computational efficiency, made it the ideal choice for predicting the difficulty ratings of Kilter Board routes. Its superior performance on both training and testing datasets suggests it can generalize well to unseen data, providing a robust and reliable approach for automating the grading of bouldering problems. Given its accuracy and computational efficiency, Histogram-based Gradient Boosting Regressor is the model of choice for deployment in this project.