Report Data Mining Project 2024/2025

Emiliano Bacci - Emanuele Buonaccorsi - Jingrui Zhu

January 6, 2025

1 Data Understanding and transformation

In this section, we will describe the data cleaning and transformation process we performed on the two datasets, including some observations on the data. Not all observations are reported here, for more detailed information refer to the notebooks, which contain all the code and many visualizations.

1.1 Duplicates

We found out that the dataset contains some duplicates, which we removed. To define what we consider a duplicate on the races, we considered a subset of attributes that we think should uniquely identify a race record: {'url_race', 'name_race', 'climb_total', 'length', 'day', 'cyclist', 'cyclist_team', 'position' 'delta'}. While on cyclist dataframe, we simply look at all the attributes.

1.2 Data Casting

We converted type *object* into *string* and *float* into *int* where is possible and makes sense, such as weight, height etc. We extracted column *day* from attribute *date*, it registered the starting time of each individual cyclist, but since the starting time is irrelevant for the final classificaltion and often contains noise, we decided to drop the time leaving only the race day.

1.3 Analysis and semantic coherence

We also performed some analysis to verify the semantic coherence of the dataset. We checked if the same cyclist has the same attributes in all records, and the results show that all the records of the same cyclist have the identical attributes. We did the same for the races.

We also want to make sure that the the inclined surface portion of race is always smaller or equal to the length of the race, which is true;

Other semantic coherence checks: we verified no cyclist has the same position in a race. We verified that 2 cyclist cannot have the same final position in a race, they can have have the same delta but the position is monotonically increasing by one.

We found out that there are some races with negative delta values. We discovered that this might be due to a particular category called ITT (Individual Time Interval). But, since the negative delta values are not coherent with the rest of the dataset, and the number of such records is very low, we decided to drop them.

We also performed some analysis to better understand some attributes. For example, we tried to understand the correlation of 'profile' with other numeric fields. After the correlation analysis (described later), we found out that it is moderatly related to 'climb_total'.

1.4 Observations

Before cleaning, we employed many visualizations to understand the datasets, and discovered that:

1. The road type attributes {'is_gravel', 'is_cobble', 'is_tarmac'} have no real significance in the dataset (see fig 1). We dropped these attributes

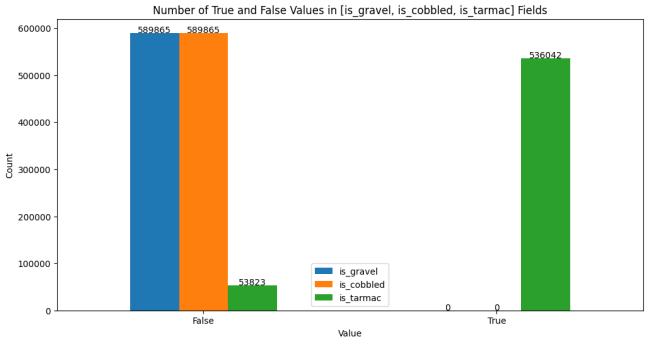

Figure 1: Road type count

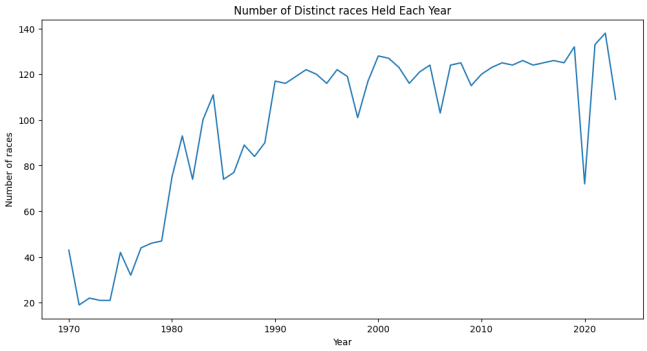

Figure 2: Race number trend over the year

2. In 2020 there was a sudden drop in the number of the race held (see fig 2), this is verified by the real event of pandemic covid19.

3. There are 2 almost distinct groups of races (see fig 3), one with high value in length, points and other with lower values. We can deduce that the first group are races of more prestigious category.

4. We dropped the attribute 'uci_points' from our dataset for it carries no contribution to the analysis, it is highly correlated to 'point' (85% using 'Pearsons' method), and only races after 2000 have this value.

More observations can be found in the notebooks.

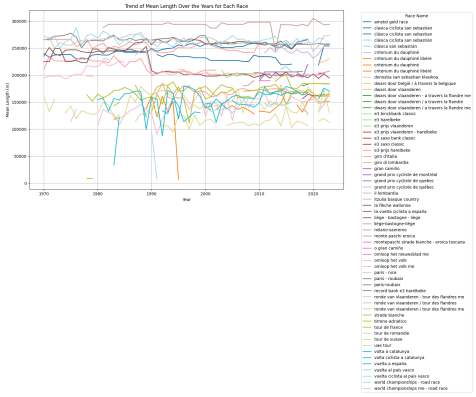

Figure 3: Trend in length of different races over the year

1.5 Handling missing values

Some attributes have most of the values missing (see fig 4), such as 'uci_points' and 'temperature'. We decided to drop them for they carry no contribution to the analysis.

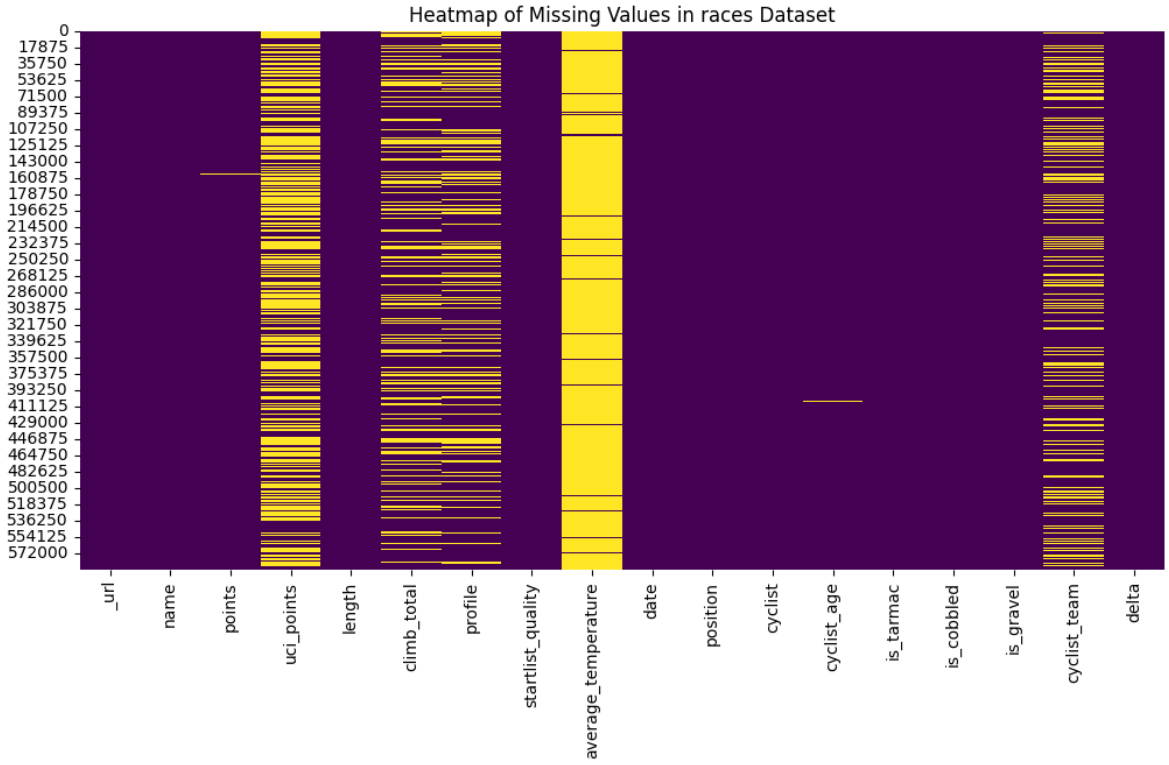

Figure 4: Missing values in the dataset

Manual fixing We found out that in cyclist dataframe, there are 1 missing value in *nationality* and 13 in *birth_year*. Because the low number, we decided to try to fill them manually by looking for the data in a reliable source and found:

1. *scott davies* with height=184, weight=66, birth year=1995, nationality=great britain
   https://www.wikiwand.com/en/articles/Scott_Davies_(cyclist)
2. *vladmir malakov* with birth year=1958
   https://www.procyclingstats.com/rider/vladimir-malakov
3. *antonio zanini* with birth year=1965
   https://www.procyclingstats.com/rider/antonio-zanini

Missing birth years There are 10 records left with missing [birth_year] value after manual searching, we decided to fill such records as such:

1. look for the first race that cyclist participated in races dataframe
2. get the mode on the age of cyclists of that year
3. reverse calculate the [birth_year] of the cyclist by subtrackting {year - mode age}

Missing weights and heights Weight and height are important and correlated attributes, we decided to fill the missing values by using the following strategy: To fill the missing values for weight and height while maintaining a normal distribution:

- For missing heights: Fill in with a random value taken from individuals of the same nationality and within a weight range of ±3 kg. If weight is also missing, fill in with a random height taken from individuals of the same nationality.

- For missing weights: Fill in with a random value taken from individuals of the same nationality and within a height range of ±3 cm.

- If there is no value found in the range, we fill with a value from the entire dataset (not only from the same nation). - For countries with less than 10 individuals, we also use the entire dataset to fill in the missing values.

Missing points Since there is only a small percentage (477 records = 0.08%) of missing values on such field, we decided to fill them using the mean of the existing values of the same race stages of the same year.

Missing ages Since we estimated the cyclists age in the cyclists dataframe, we can now simply calculate the missing values.

Data understanding has been repeated after cleaning to verify the correctness of the cleaning process.

1.6 Data Cleaning and Outlier detection

1.6.1 Cyclist - Weight Height

Since these attributes follow a normal distribution, we can easily detect (extreme) outliers by using the Inter-quartiles and drop them.

1.6.2 Races - Points Races

The graph 5 shows that the average points for each year before 1977 are unusually high, so we drop races between 1970 and 1977, for they would disrupt the analysis of the dataset.

The fig 6 shows the average race length of each different race over the year, as already mentioned in the section *before cleaning*, there are 2 sub groups of races. And upon observing, we noticed that longer the race, higher is the points awarded to the cyclist.

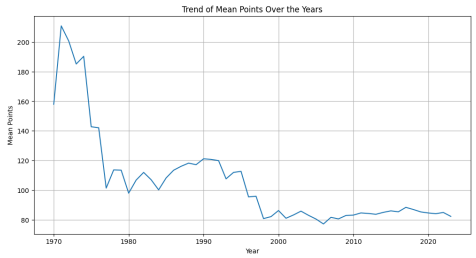

Figure 5: Average awarded points over the year

Figure 6: Average individual race length over the year

1.6.3 Races - Cyclist age

We noticed the presence of a cyclist of age 13 at the time of the race, it is clearly an error and/or outlier, so it has been removed. A cyclist of age over 50 at the time of the race, for it is an extreme value, so it has been removed.

1.6.4 Cyclists cleaning

We found out that there are 397 cyclists who are registered in the dataset but have never competed in a single race, they are not interesting to analyze, so we drop them.

Because we updated the cyclist dataframe by removing some of them, we also need to remove the correlated records in races dataframe to maintain consistency.

1.6.5 New Features

We added some new features to each cyclist record, in order to better capture the characteristics of the cyclist and of their previous performance.

1. BMI - Body Mass Index
2. races_participated (total number of races partecipated by the cyclist)
3. average_race_length (average length of races partecipated by the cyclist)
4. average_climb_length (average length of races partecipated by the cyclist on inclined surface)
5. intensity
6. average_intensity

In figures 7 and 8 we show the distribution of the new features and the boxplots of the new features. We can make some observations:

- Many races take place on an entirely flat surface.
- Many cyclists only participate to races on a flat surface, as their average_climb_total is 0.
- The number of races participated by a cyclist is highly skewed, with most cyclists participating in a small number of races, with some outliers participating in more than 300 races.
- The average delta follows a quasi-normal distribution, with a considerable number of cyclist arriving at the finish line with a delta of 0 or around 0, meaning they arrived at the same time as the first cyclist.

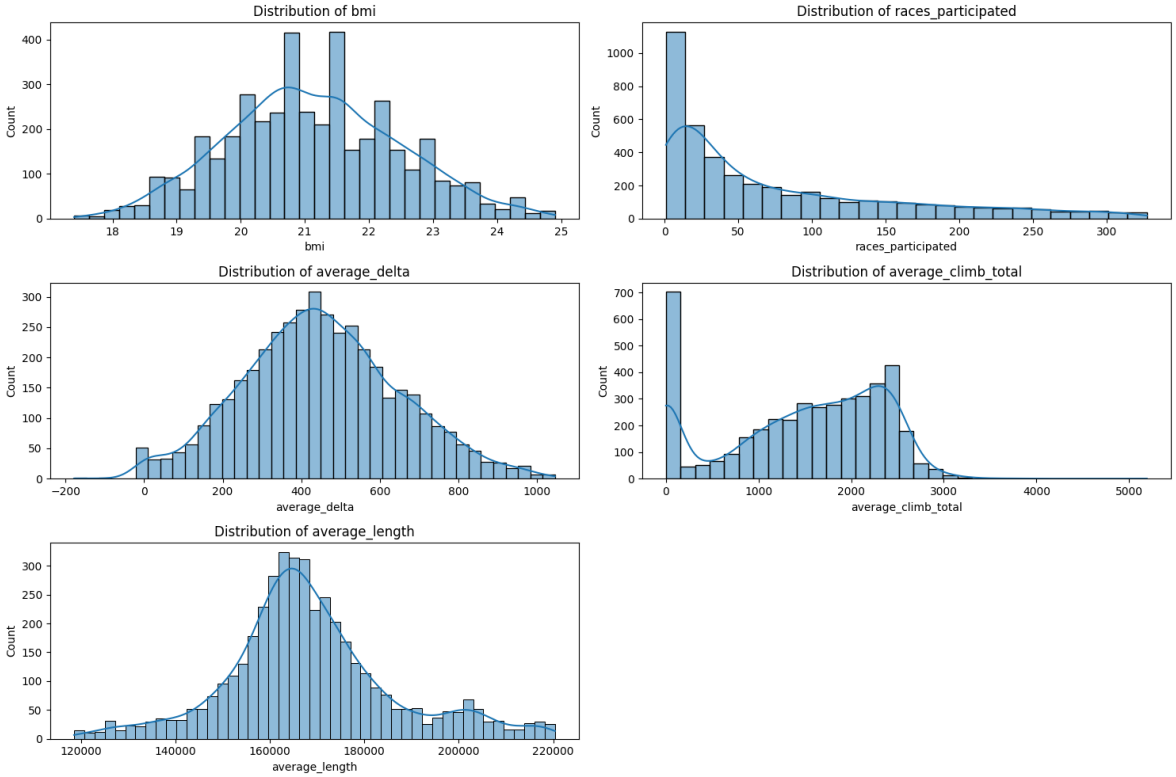

Figure 7: Distribution of new features

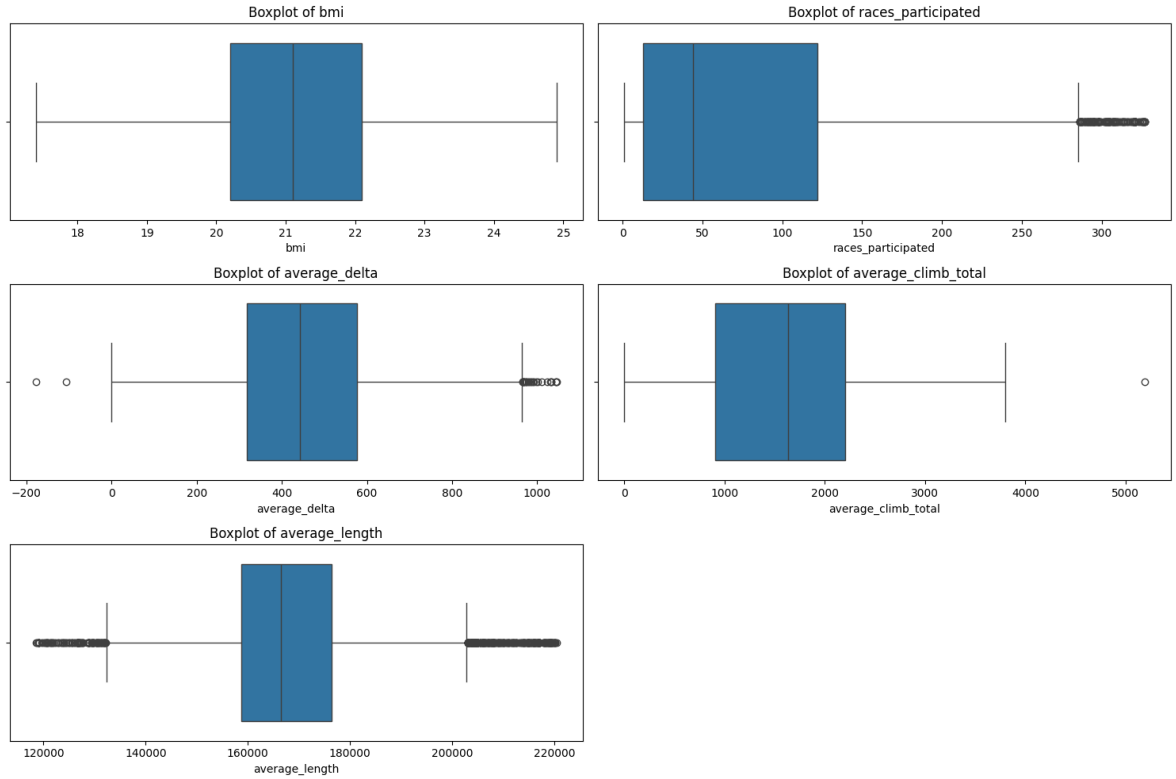

Figure 8: Boxplots of new features

1.7 Attributes and correlation

Attribute	Value Type	Description	Data Type
url_cyclist	string	Unique identifier of the cyclist	Categorical
name_cyclist	string	Name of the cyclist formed by name and surname	Categorical
name_cyclist	string	Name of the cyclist formed by name and surname	Categorical
nationality	string	Nationality of the cyclist	Categorical
birth_year	int	Birth year of the cyclist	Numerical
weight	int	weight of the cyclist, measured in kg	Numerical
height	int	Height of the cyclist, measured in cm	Numerical
bmi	float	BMI of the cyclist, given by relationship between weight and height	Numerical
races_participated	int	The number of registered races the cyclists has participated	Numerical
average_race_length	int	The average race length the cyclists has run, measured in m	Numerical
average_climb_length	int	The average race length on inclined surface the cyclist has run, measured in m	Numerical
average_intensity	float	The average inclination of the road the cyclist has run on	Numerical

Table 1: Cyclists dataframe attributes after cleaning

Attribute	Value Type	Description	Data Type
url_race	string	Unique identifier of the race formed by name/year/stage	Categorical
name_race	string	Name of the race	Categorical
points	int	Points awarded to the cyclists	Numerical
length	int	the Total length of the race	Numerical
climb_total	int	The part of the race on inclined surface	Numerical
profile	int	Difficulty level of the race	Numerical
startlist_quality	int	How strong a cyclist is	Numerical
position	int	The order in which the cyclist arrived at the finish line	Ordinal
cyclist	string	The name of the cyclist participating the race	Categorical
cyclist_age	int	The age of the cyclist at the moment of the race	Numerical
cyclist_team	string	The team which the cyclist belongs to	Categorical
delta	int	the time difference the cyclist arrives at the finishing line after the first racer	Numerical
day	date	The day of the race	Categorical
intensity	float	The inclination of the part of inclined road	Numerical

Table 2: Races dataframe attributes after cleaning

After cleaning, we performed a correlation analysis on the dataset to understand the relationship between the attributes and decide which attributes to use in the analysis - fig 9.

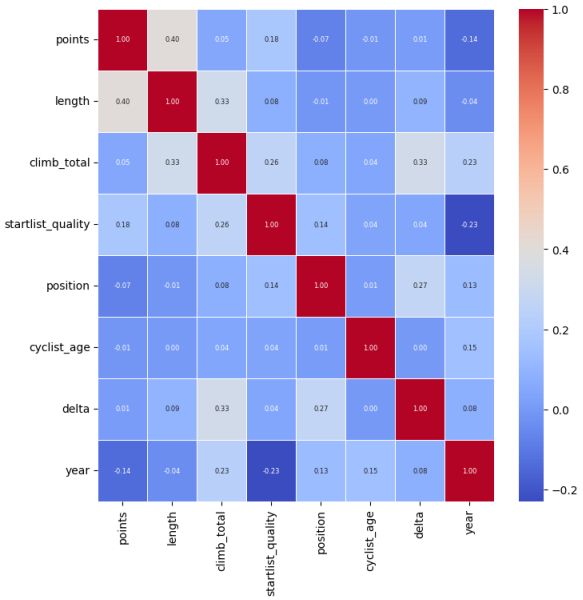

Figure 9: Correlation matrix on attributes of races

2 K-Mean analysis

The objective is trying to understand how cyclists's physical attributes(weight and bmi) may influence their choice of races (flat, inclined or both).

We selected features [bmi, weight, average climb length, average race length, average intensity] for they could interprate our goal. We do not want to rank cyclists for there are point-systems already being used officially, we simply want to understand where the physical attribute of a cyclist could influence their choice of race.

We use min-max scaler to normalize the features, because the most of the selected feature have a highly skewed distribution.

For the calculation of the optimal value of clusters k, we used SSE score, Silhouette score and Davies-Bouldin score. Silhourtte measures how well each data point fits within its assigned cluster compared to other clusters. Davies-Bouldin assesses the compactness and separation of clusters in a dataset. Lower values indicate better clustering. After the consideration, we choose k=3. The resulting silhouette score is rounded to 0.3

In fig 10 shows the scatter plot of attributes of clusters

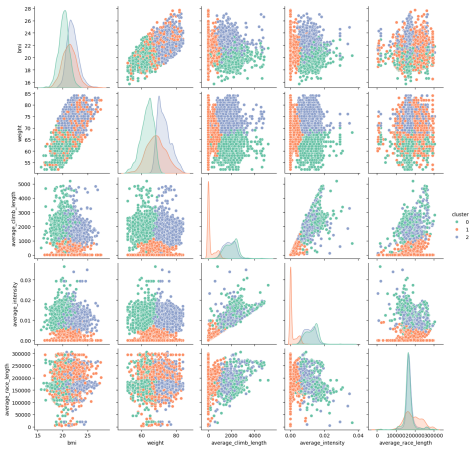

Figure 10: Scatter plot of attributes of clusters

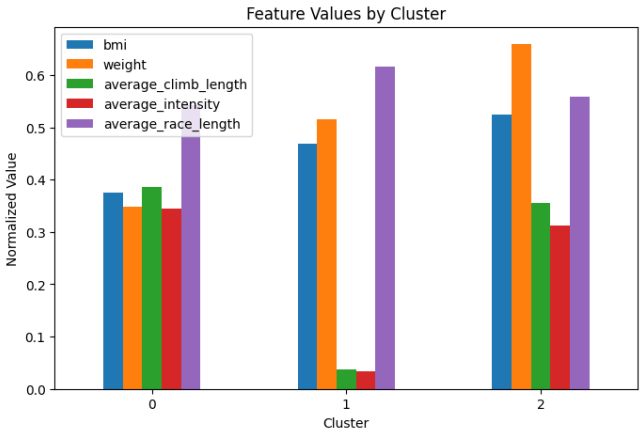

Figure 12: Cluster visualization

In fig 11 shows the importance of each feature used in our analysis, this is calculated through the use of random forest.

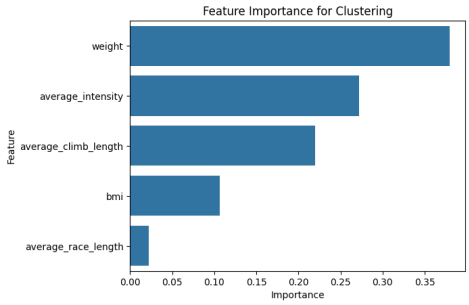

Figure 11: Feature importance in Kmean

2.1 Analysis

We first ignore the 'average_race_length' feature for it is not important in our analysis and also, it shows the similar range of values in all clusters. In fig 12 shows the centroids of attributes in each cluster:

1. cluster 0 has low bmi weight but high on inclined surface indicators
2. cluster 1 has high bmi weight but low on inclined surface indicators.
3. cluster 2 has high bmi weight and also high on inclined surface indicators

We observe that:

1. cyclists with lower weight have a higher participation (distance indicators) on inclined surface race.
2. cluster 1 are cyclists that don't compete on inclined surface.
3. cluster 2 are cyclists that can 'feel ease' on both inclined and flat surface

3 DBSCAN analysis

The goal is to understand the preference in the road type (flat or inclined or both) according to their frequency in competing.

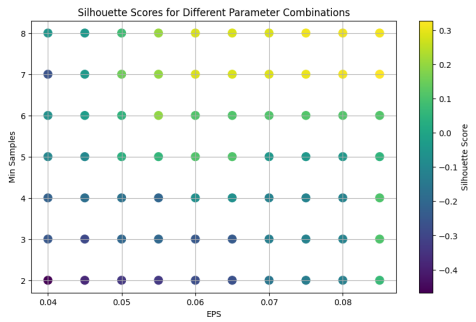

Figure 13: Silhouette score of eps/min sample combinations

We select features ['races participated', 'average race length', 'average climb length', 'average intensity'] and again, use min-max scaler to normalize them.

The method requires 2 hyperparameter, eps and min sample. The eps is the maximum radius of the neighborhood around a point to be considered part of the same cluster. And we try to find the value by using both KNN method and Knee method where in KNN, the 'imaginary' min sample is 2*number of features while in Knee, the distance between pairwise points are calculated using euclidean. We find out that the best eps ranges in [0.04, 0.08], so we attempted to find the best combination of eps and min sample by varing eps in its range and varing min sample in [2, 9], and use silhouette score as metrics.

In fig 13 shows the silhouette score of the combination, according to the results, combinations with eps=0.06, 0.07, 0.08 and min sample=7, 8 produce almost equivalently good results, so we decided to choose arbitrary epq=0.07 and min sample=7. And also, these combinations don't produce a large nor too little number of clusters.

3.1 Analysis

With the parameters, we proceeded with DBSCAN on the datasets and produced 4 clusters.

1. cluster 1 functions as the baseline for the dataset, where majority of cyclists belong to,they have consistant preference in road type

2. cluster 0 has 12 members, acting like a singularity, after analyzing the results, we discover that this cluster has only cyclists that have run 1 specific race they are low in the races length, both on flat and inclined surface. But the race they run on has a high(er) percentage of inclined surface, this could impliy that they were a subset of specialized cyclists for the race.

3. cluster 3 has 14 members, with low results on all features, we can deduce that the cyclists in this cluster are less experienced racers.

4. cluster 2 has 36 members, even with low participation, they achieve high results on other feature, we can categorize them as above average cyclists experienced on different type of races.

5. cluster -1 (outliers) with 77 members, are a mix of extremely experienced cyclists with high number of race participation but below average performance, underperforming cyclists, elits cyclists and data-anomalies.

4 Hierarchical Analysis

4.1 Objective

The goal of this analysis is to understand the types of cyclists based on their race metrics (average climb length, average race length), experience (number of races participated), and physical attributes (weight). We aim to categorize cyclists into clusters that reflect their characteristics and preferences in race types.

4.2 Data Preprocessing

To minimize noise, we filtered the dataset by removing extreme values, keeping only records within the 5th to 95th percentiles of the `races_participated` feature.

The selected features for clustering are:

- `weight`
- `races_participated`
- `average_race_length`
- `average_climb_length`
- `average_intensity`

We used MinMaxScaler to normalize these features due to their highly skewed distributions.

4.3 Clustering Methodology

Using hierarchical clustering, we calculated the Euclidean distance between data points and employed the complete linkage method. The dendrogram in Figure 14 illustrates the hierarchical clustering structure.

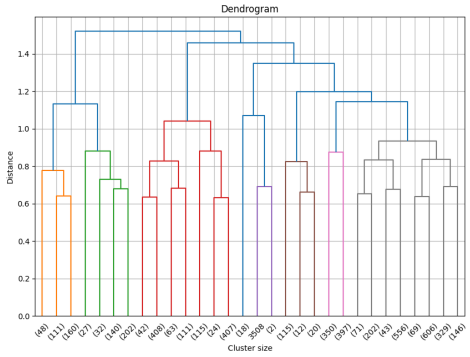

Figure 14: Dendrogram of Hierarchical Clustering

We chose to cut the dendrogram at a distance of 1.2, resulting in four clusters.

4.4 Cluster Analysis

The characteristics of each cluster are summarized below, based on the mean values of the selected features within each cluster.

- **Cluster 0**: Cyclists with moderate `weight`, `races_participated`, and `average_climb_length`. They represent average cyclists experienced in both flat and inclined races but not as seasoned as Cluster 2.
- **Cluster 1**: Cyclists with the lowest `weight` and highest `average_climb_length`. They are skilled in inclined surface races.
- **Cluster 2**: Cyclists with the highest `races_participated` and high values for other metrics. These are experienced cyclists proficient in both flat and inclined races.
- **Cluster 3**: Cyclists with the lowest values across all metrics, indicating they are less experienced racers.

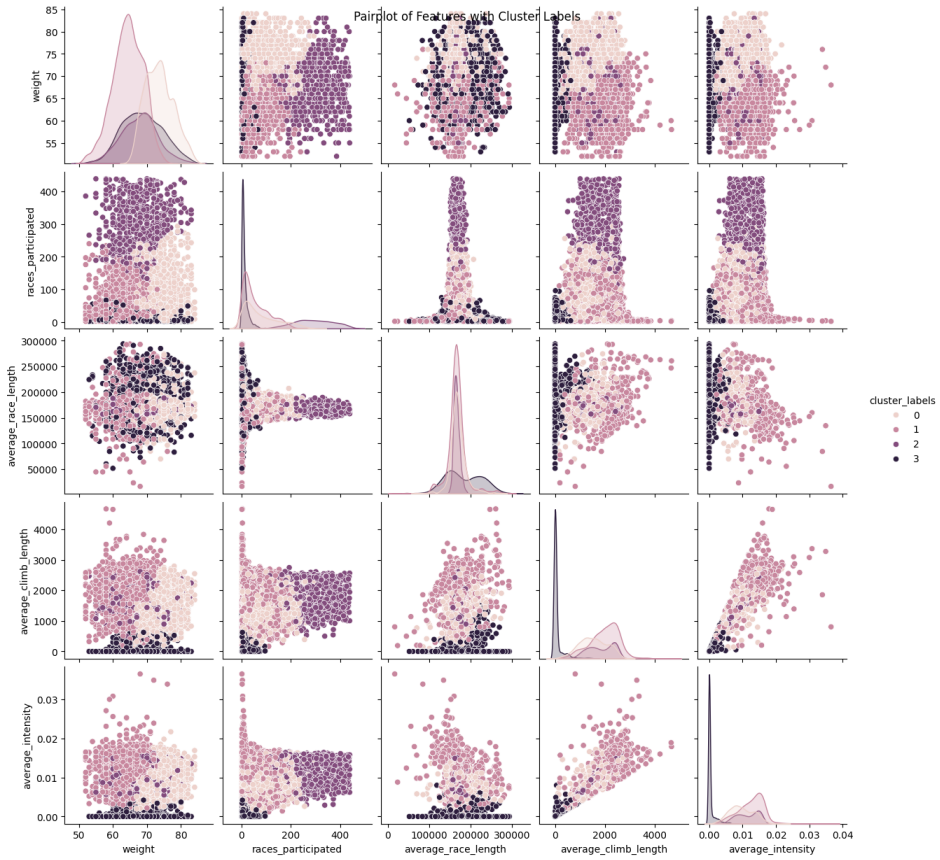

Figure 15: Cluster Visualization by Feature

4.5 Feature Importance

Using a Random Forest classifier, we evaluated the importance of each feature in distinguishing between clusters. The results, shown in Figure 16, indicate that `races_participated` and `weight` are the most critical features.

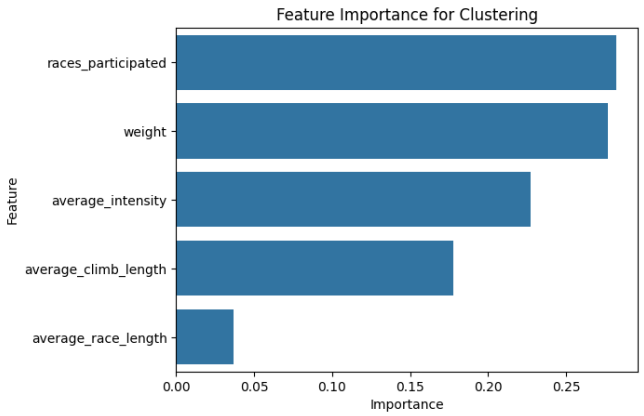

Figure 16: Feature Importance for Hierarchical Clustering

4.6 Observations

- **Cluster 3** includes cyclists with minimal race participation and low physical metrics, suggesting they are less experienced.

- **Cluster 2** features the most experienced cyclists, excelling in various race types.
- **Cluster 1** highlights cyclists specializing in races with significant climbs.
- **Cluster 0** represents average cyclists proficient in multiple race types but less seasoned than those in Cluster 2.

The silhouette score for this clustering is 0.26, indicating moderate clustering performance.

4.7 Conclusion

The hierarchical clustering analysis effectively categorized cyclists based on physical attributes and race metrics, revealing insights into their preferences and specialization in race types.

5 Prediction task

We evaluated five different types of prediction models to determine the most effective approach for our dataset. The dataset was divided into training, validation, and test sets. As per the project requirements, the test set included races from 2022 onward. This test set was used only once, after the models were selected, to ensure unbiased performance evaluation.

5.1 Data Preprocessing

5.1.1 Label Calculation

For each record in the dataset, we calculated the `is_top_20` label, indicating whether a cyclist ranked in the top 20. We observed that only 14.8% of the records had `is_top_20=true` 17, creating a significant class imbalance. To address this, we undersampled the majority class to achieve a balanced 50%/50% class distribution 18.

We observed that the itorduction of the undersampling technique significantly improved the recall for the minority class (`is_top_20=true`), which would otherwise have been underrepresented.

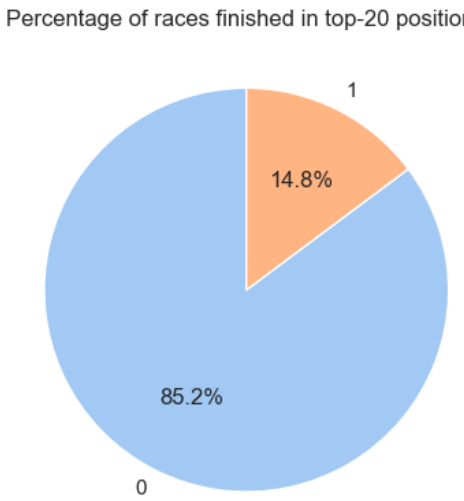

Figure 17: Before undersampling

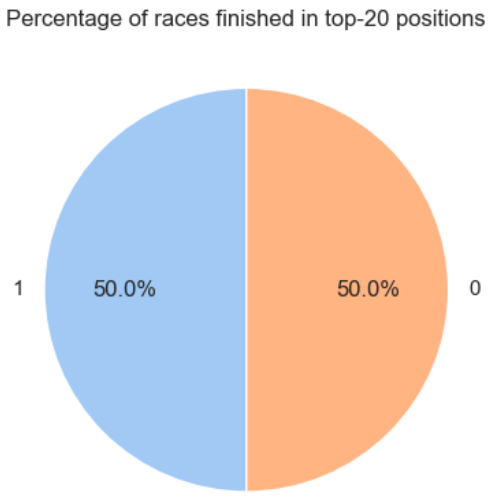

Figure 18: After undersampling

5.1.2 Discretization and Feature Selection

We applied one-hot encoding to discretize the categorical data, which improved the model's performance compared to numerical discretization methods. Additionally, we selected the relevant features for prediction and excluded those that were either irrelevant or features that should be unavailable

until the race finishes. This feature selection process ensured that the models focused only on the most useful information.

5.2 Model Evaluation

Each model was evaluated using precision, recall, F1-score, and support metrics on both the training and validation sets. We used visualizations such as confusion matrices and ROC curves for a clearer understanding of each model's performance.

5.2.1 K-Nearest Neighbors (KNN)

We performed a grid search to optimize the parameters for `k`, the distance metric, and the weight strategy. The best parameters found were:

```
{'weights': 'distance', 'n_neighbors': 20, 'metric': 'manhattan'}
```

Performance on the validation set was as follows:

	precision	recall	f1-score	support
Not Top-20	0.77	0.72	0.74	9172
Top-20	0.73	0.78	0.76	9195
accuracy			0.75	18367
macro avg	0.75	0.75	0.75	18367
weighted avg	0.75	0.75	0.75	18367

We observe that the KNN model performed well, with balanced precision and recall.

5.2.2 Naive Bayes

Naive Bayes was tested on the training set, and its performance is summarized below:

Report of the performance on the VALIDATION set:

	precision	recall	f1-score	support
Not Top-20	0.62	0.65	0.64	9172
Top-20	0.63	0.61	0.62	9195
accuracy			0.63	18367
macro avg	0.63	0.63	0.63	18367
weighted avg	0.63	0.63	0.63	18367

Observation: The Naive Bayes model struggled with classifying the set correctly, maybe due to its strong assumptions of feature independence.

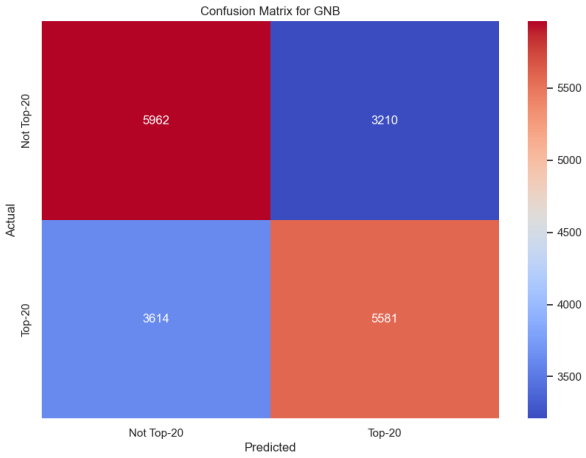

Figure 19: Confusion matrix for Naive Bayes

5.2.3 Rule-Based (RIPPER)

Given the high computational cost, we opted for `RandomizedSearchCV` for parameter tuning. The best parameters identified were:

```
{'prune_size': 0.3, 'k': 1}
```

Performance on the training set:

```
Report of the performance on the VALIDATION set:
              precision    recall  f1-score   support

  Not Top-20       0.55      0.97      0.70      9172
      Top-20       0.87      0.23      0.36      9195
```

Observation: While the RIPPER algorithm achieved high precision for the minority class, its recall was very low.

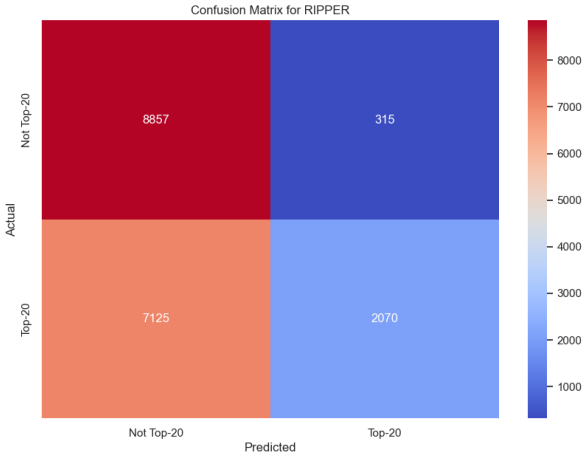

Figure 20: Confusion matrix for RIPPER

5.2.4 Decision Trees

We conducted a grid search over many parameter combinations. The training process was fast, and the best parameters found were:

```
{'criterion': 'entropy', 'max_depth': 7, 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': '
```

Performance:

```
Report of the performance on the VALIDATION set:
              precision    recall  f1-score   support

  Not Top-20       0.67      0.73      0.70      9172
      Top-20       0.70      0.64      0.67      9195

    accuracy                           0.68     18367
   macro avg       0.68      0.68      0.68     18367
weighted avg       0.68      0.68      0.68     18367
```

Observation: Decision Trees showed a good balance between precision and recall, but its performance was not as high as KNN. there might be potential room for improvement in parameter tuning.

5.2.5 Neural Network (Keras)

The Keras-based neural network model was structured as follows:

```
keras_model = tf.keras.Sequential([
    Input(shape=(train_set_scaled.shape[1],)),
    Dense(128, activation='relu', kernel_regularizer=l2(0.001)),
    Dropout(0.3),
    Dense(64, activation='relu', kernel_regularizer=l2(0.001)),
    Dropout(0.3),
    Dense(32, activation='relu', kernel_regularizer=l2(0.001)),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])
keras_model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=[tf.keras.metrics.BinaryAccuracy(), tf.keras.metrics.Precision(), tf.keras.metrics.Recall
)
```

We selected this architecture because it is well-suited for binary classification tasks. We experimented with various architectures and parameters, and this one gave the best results. Although we chose the 'binary-cross-entropy' loss function, our evaluation considered multiple factors 21. We manually tuned some parameters for optimal performance.

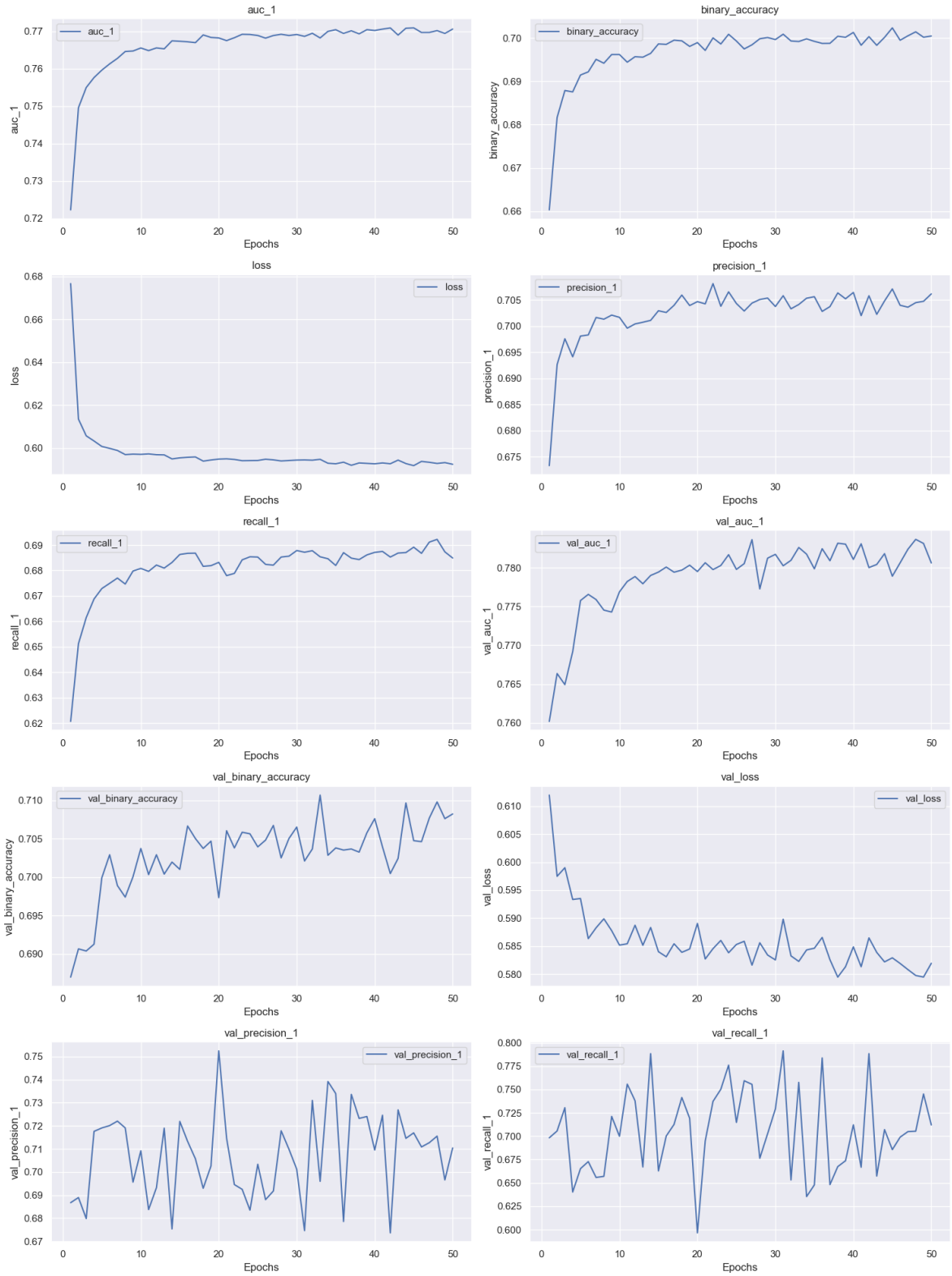

Figure 21: Some of the metrics taken into consideration while tuning the Keras model

Performance on the training set:

	precision	recall	f1-score	support
Not Top-20	0.72	0.72	0.72	9172

```
Top-20          0.72        0.73        0.72        9195
```

It shows a good precision and recall, balanced for both classes.

5.3 Model Comparison

We compared the models using ROC curves to visualize their performance:

- KNN (AUC = 0.75)
- GNB (AUC = 0.63)
- RIPPER (AUC = 0.60)
- Decision Tree (AUC = 0.68)
- Keras (AUC = 0.72)

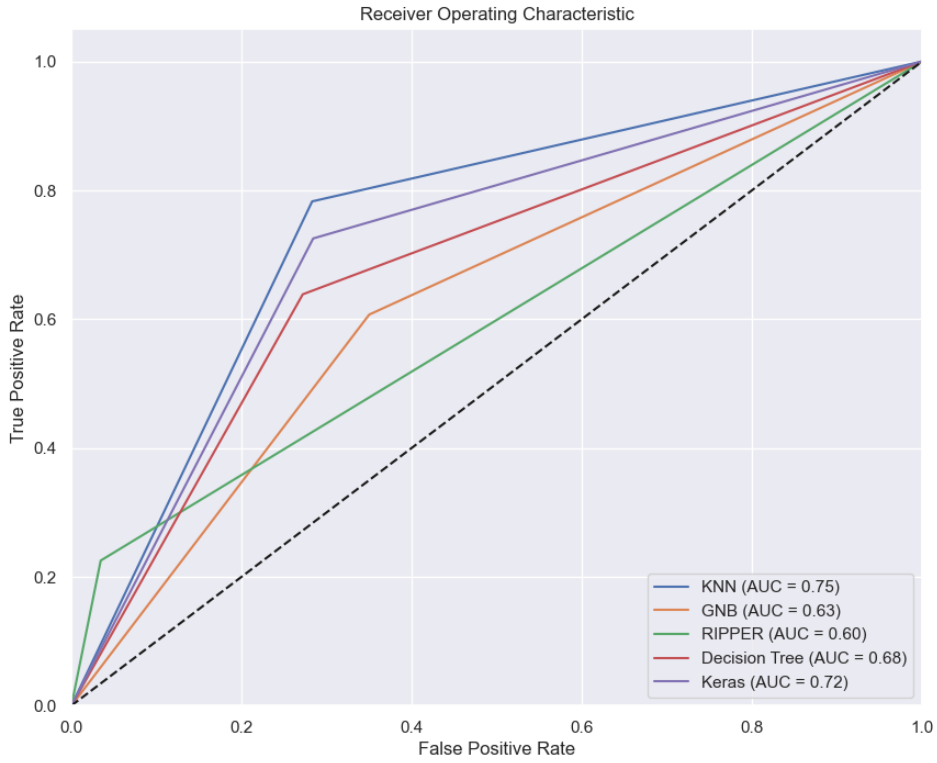

Figure 22: ROC for the validation set

The KNN model had the highest AUC, followed by the neural network. RIPPER, with the lowest AUC, struggled to handle the complexity of the dataset.

6 Test Set Results

To conclude, we evaluate the best 3 models on the test set. To avoid introducing bias, the test set has been used only once.

The performance of the Decision Tree model dropped significantly in the test set, highlighting potential overfitting during training.

KNN and the neural network generalized better than the decision tree, though the recall on the minority class dropped.

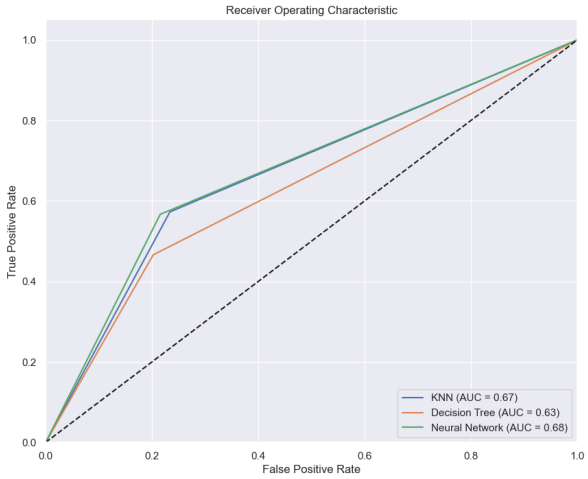

Figure 23: ROC on the test set

7 Explanation

7.1 Introduction

In this section, we provide explanations for some of the machine learning models we developed and selected in the previous task We will explore aspects such as feature importance, rule explanations, and counterfactual instances.

7.2 Overview of Selected Models

We chose to focus on two distinct models: the Decision Tree (DT) and the Keras Neural Network (NN). These models were selected because they performed well in the previous task and offer an interesting contrast in terms of complexity and interpretability. The Decision Tree is straightforward and easy to interpret, while the Neural Network is more complex and harder to interpret.

7.3 Explaination for the Decision Tree Model

7.3.1 Feature Importance Analysis

Built-in Feature Importance Scikit-learn's Decision Tree model provides a built-in feature importance metric, which quantifies the contribution of each feature to the model's decision-making process. By extracting and visualizing these scores, we were able to identify the most influential features. As shown in Figure 24, `average_delta`, was by far the most important feature in the tree's decision making. This makes sense, since the previous performances of the cyclist are a good indicator to predict the future ones.

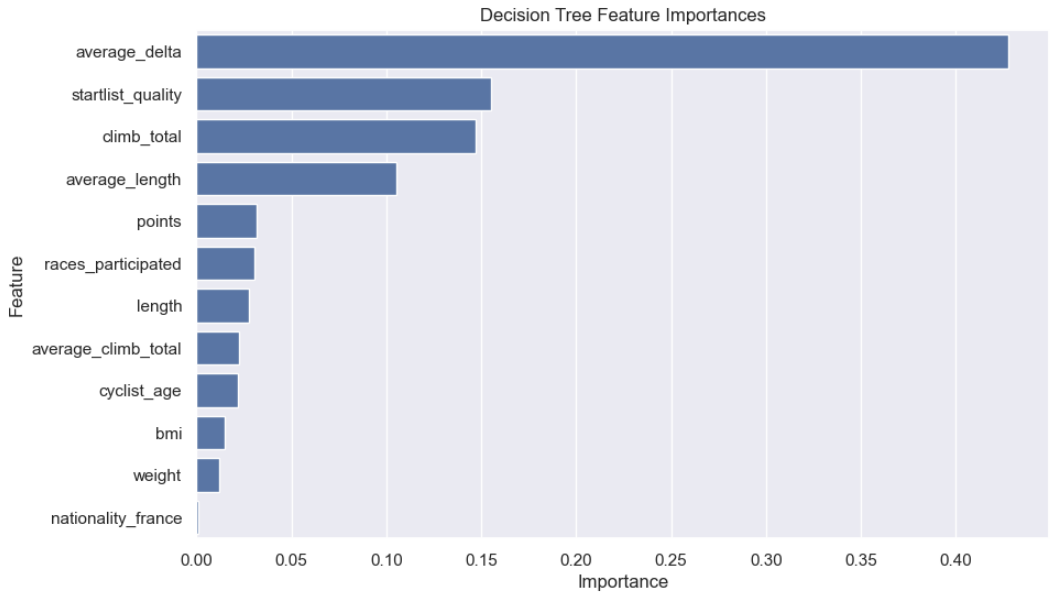

Figure 24: Feature Importance of Decision Tree Model

Using SHAP Values Even though the built-in feature importance metric already provides clear information about the importance of each feature, we wanted to also test the SHAP method to see if it would provide the same results.

As expected, the results are really similar 25.

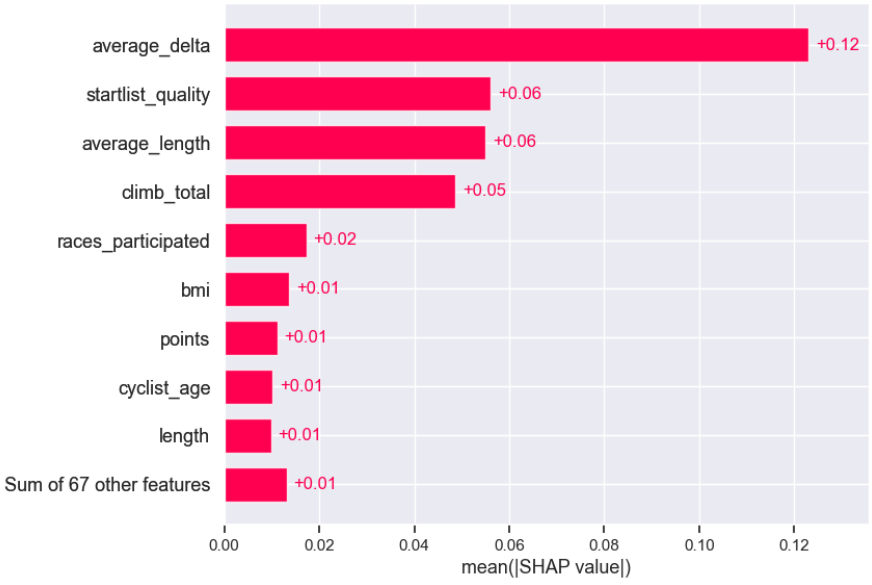

Figure 25: SHAP Feature Importance of Decision Tree Model

SHAP also enables us to visualize waterfall plots for individual predictions, which show how each feature contributes to the final prediction. Figure 26 shows the waterfall plot for a single prediction. The plot shows that the prediction was made by adding the contributions of each feature, with the most influential features at the top of the plot.

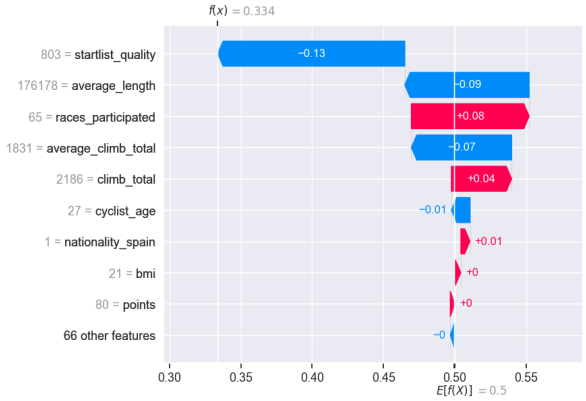

Figure 26: Waterfall plot on a particular instance

To evaluate the fidelity of SHAP, we also compared it to the built-in importance metric, getting really similar results 27

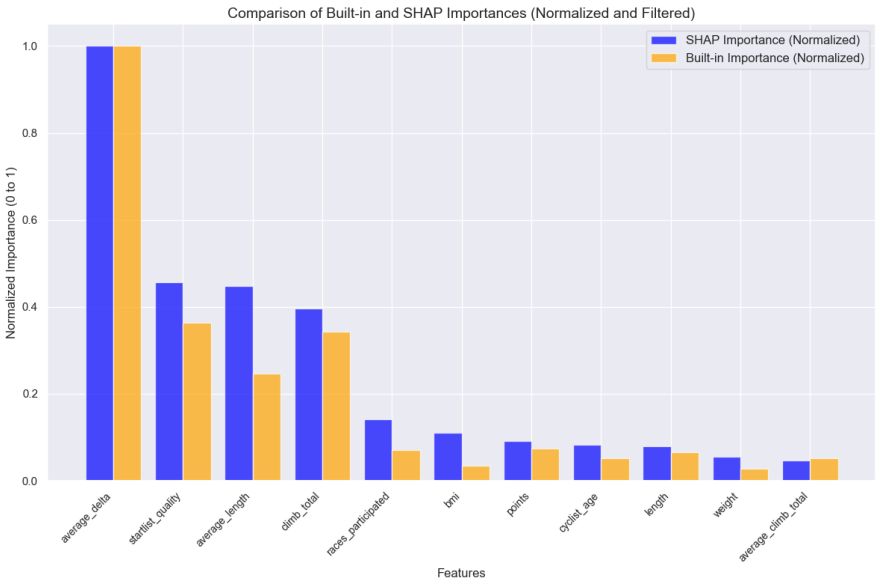

Figure 27: Comparison of SHAP and built-in importance metric for tree models

7.3.2 Explaining Model Behavior

7.3.3 Rule Explanation

Decision Trees are a transparent model, so we analyzed the tree structure and the decision rules at each node. We can observe it by looking at the tree visualization 28.

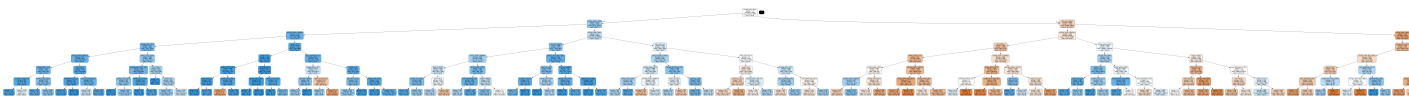

Figure 28: Enter Caption

The tree is composed by 127 nodes, resulting in a big number of rules to be examined to comprehend the model as a whole. By examining individual instances instead, we can focus on the rules involved in the decision-making of the specific instance.

We were able to identify how specific thresholds in key features, such as `climb_total`, influenced the final prediction, confirming the importance of these features.

For example, let's take a particular instance (instance 0, which was classified as is_top_20=1). The rules involved in its classifications (i.e.: the path followed from the root to the leaf) are:

- `average_delta` >334.5
- `climb_total` <= 2660.0
- `average_length` >166136.5
- `startlist_quality` <= 848.5
- `average_delta` >464.5
- `startlist_quality` >428.5
- `races_participated` >128.5

Leaf node 174 was reached, with a prediction of `[[0.44924285, 0.55075715]]`.

7.3.4 Counterfactual Instances

We used DiCE to generate counterfactuals, which help us understand how small changes in input features could alter predictions. By making minimal adjustments to the values of key features, we observed how the predicted class would change. This process revealed that certain features had a more substantial impact on predictions, providing valuable insights into the model's sensitivity to specific inputs.

Original Instance (Outcome: 0) Here is the original instance that resulted in a prediction of 0 (not in the top 20):

points	length	climb_total	startlist_quality	cyclist_age	weight	bmi	races_participated	average_climb_total
225.0	196200.0	1652.0	727.0	24.0	71.0	20.3	53.0	2143.0

Diverse Counterfactual Set (New Outcome: 1) By generating counterfactuals, we adjusted key features to explore how minimal changes in inputs could shift the outcome. The table below shows three counterfactual instances that resulted in a prediction of 1 (top 20). These changes highlight which features have a significant impact on improving the prediction. For instance, small increases in `bmi` and `races_participated` were sufficient to change the predicted outcome from 0 to 1 in certain cases.

points	length	climb_total	startlist_quality	cyclist_age	weight	bmi	races_participated	average_climb_total	av
–	–	–	–	–	–	24.6	–	–	
–	–	–	–	–	–	21.0	315.3	–	
–	–	–	–	–	–	24.7	–	–	

We observe, in particular, that a change in the bmi attribute would change the prediction, according to these counterfactuals. In fact, in each of these counterfactuals, the `bmi` value was adjusted slightly.

7.4 Explanation for the Neural Network Model

7.4.1 Feature Importance

Understanding the importance of features in the neural network model can be more challenging due to its complexity, but several techniques can be used to understand which features have the most significant impact on predictions.

Examples on Instances: SHAP Waterfall Plot Analysis One effective method for explaining the model's decisions is by using SHAP (SHapley Additive exPlanations) values. SHAP provides a unified approach to explain the contributions of each feature to the model's output for individual instances.

For instance, a SHAP waterfall plot shows how the prediction is broken down by individual feature contributions. This method allows us to interpret the influence of each feature on the model's decision. In Figure 29, we show the waterfall plot for a specific instance. The plot reveals that features such as `climb_total`, `startlist_quality`, and `races_participated` are the most influential in driving the final prediction.

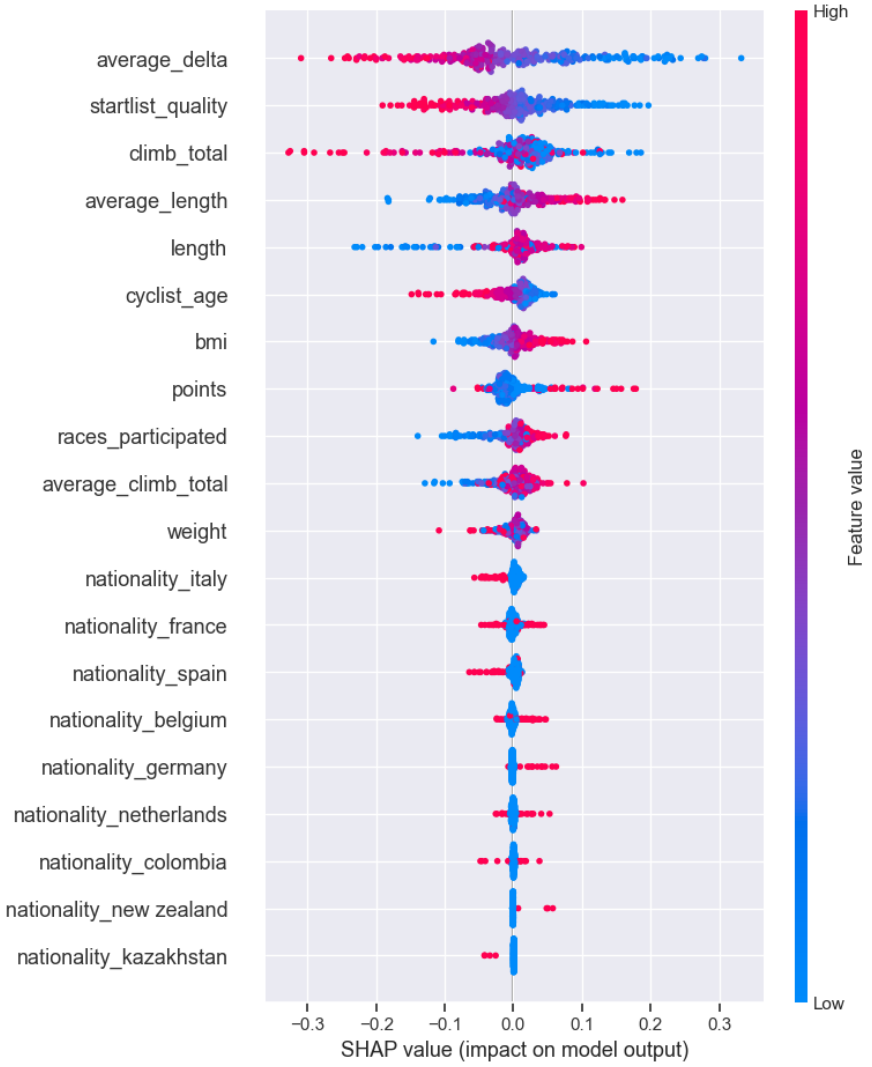

Figure 29: SHAP Waterfall Plot for Neural Network Model

This method gives a clear picture of how each feature contributes to a prediction, allowing us to make transparent conclusions about the model's behavior for specific instances.

A summary plot of SHAP values for all instances can also provide a global view of feature importance. This analysis can help identify which features are consistently influential across the dataset and provide insights into the model's overall behavior.

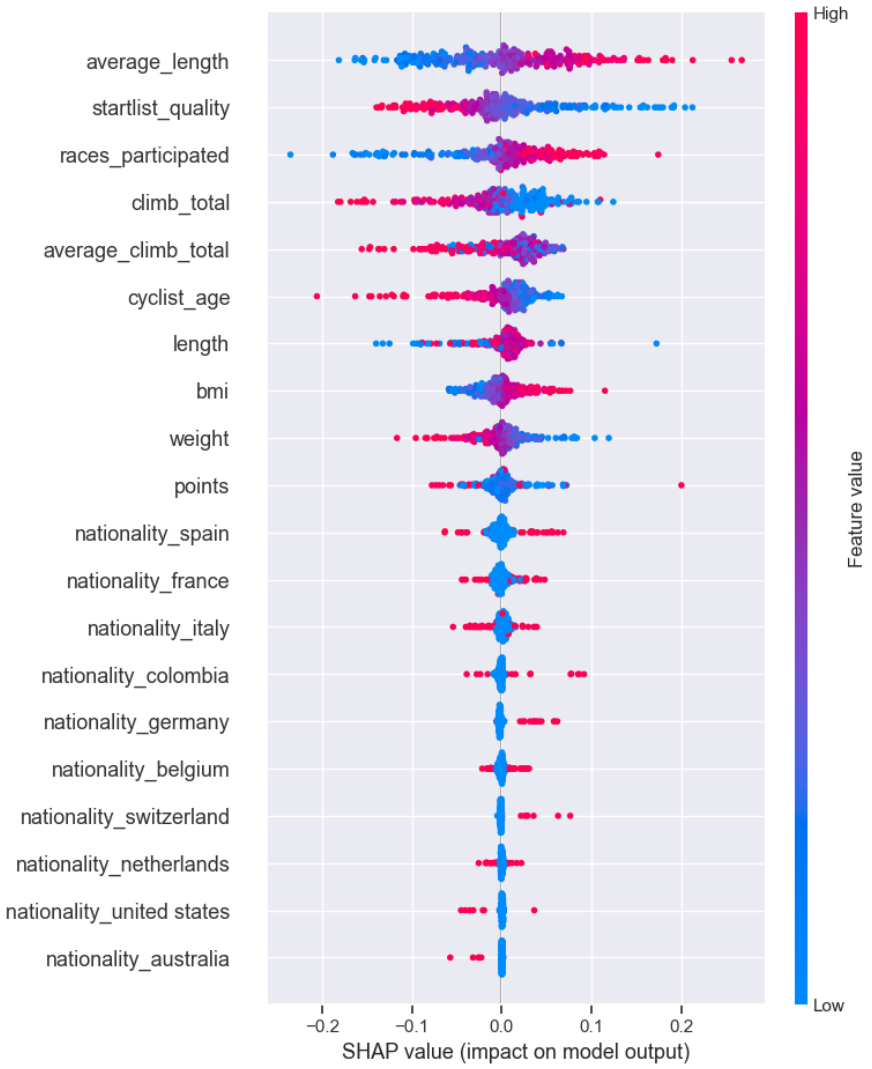

Figure 30: SHAP Summary Plot for Neural Network Model

In addition to SHAP, we also used LIME (Local Interpretable Model-Agnostic Explanations) as a surrogate model to explain the neural network's decisions. LIME generates a local interpretable model for a given prediction by approximating the complex neural network with a simpler, interpretable model (such as a linear model) in the vicinity of the prediction.

The LIME scores obtained are comparable to SHAP values and provide a similar interpretation of feature importance.

7.4.2 Rule Explanation

Neural networks, unlike decision trees, are not inherently interpretable, as they rely on complex layers of interconnected nodes. Howe

While the model itself doesn't have explicit rules like a decision tree, the analyses we performed and the counterfactual instances suggest that specific thresholds and combinations of features, such as a high `races_participated` value, an optimal `bmi` or a small `average_delta`, can significantly shift the predicted outcome.

We attempted to use the `LIMxailib.explainers.lore_explainer` to extract rules from the neural network model. However, the computation time for a single instance was excessively long, even though we tried different configurations, so we were unable to extract rules using this method.

7.4.3 Counterfactual Instances

We also explored counterfactual explanations for the neural network model using DiCE to understand how slight changes in feature values could alter predictions. Just as with the Decision Tree model, counterfactuals provide a valuable tool for interpreting the decision boundaries of the neural network.

Observing the Counterfactuals One of the counterfactuals generated for the neural network model showed that small adjustments in features like `bmi` and `average_delta` can change the predicted outcome significantly. For instance, increasing the `bmi` value to around 24.5 and `races_participated` to over 150 resulted in a change from a prediction of 0 (not in top 20) to 1 (top 20).

We observed that while the neural network model is more complex than decision trees, its sensitivity to key features like `races_participated` and `average_delta` aligns with the trends seen in the decision tree. This provides useful insights into the features that influence the predictions of cycling performance in both models.

7.5 Final Observations

The patterns identified in the model outputs largely align with what we learned through data exploration and clustering analysis, particularly the importance of `average_delta`, which also aligns with intuition about the impact of previous performance on future results.

However, some models, particularly the neural network, uncovered new patterns that were less evident in earlier analyses. For example, excessive race participation was found to negatively impact performance.

While most predictions align with expectations, some anomalies deserve further investigation. For example, there are many cases where a high-quality start list was associated with a lower likelihood of top finishes. This might suggest that the model may disproportionately focus on race-specific characteristics rather than cyclist-specific attributes. This limitation may come from a lack of cyclist-focused features in the dataset.

Overall, the models performed reasonably well, given the complexity of the prediction task, but there is room for improvement. These observations highlight the potential benefit of integrating new features, either from external data sources or through the creation of novel attributes. Enhancing the dataset with cyclist-specific charateristics could shift the model's focus from race attributes to the individual cyclist, potentially leading to more accurate predctions.