UNIVERSITÀ DI PISA

DIPARTIMENTO DI INFORMATICA

Data Mining Project

Authors:

Francesco Benocci (602495)

Pamela Di Clemente (674029)

Leonardo Manneschi (599933)

ANNO ACCADEMICO 2024/2025

Contents

Chapter 1

Data Understanding

In this chapter, we examine the provided CSV files (*cyclist.csv* and *races.csv*) separately to maintain clarity and consistency. Any type-casting will be addressed in Task 2 during the data transformation phase. The code used for this analysis is available on GitHub at this repo.

1.1 Dataset Overviews

1.1.1 Cyclist Dataset (*cyclist.csv*)

We begin by detailing the attributes of the **cyclists** table, including their semantics and inferred types, as shown in Table 1.1.

Table 1.1: Attributes Overview for *cyclist.csv*

Attribute Name	**Semantics**	**Type**
`_url`	Identifier of the cyclist	Object
`name`	Full name	Object
`birth_year`	Year of birth	Float64
`weight`	Weight	Float64
`height`	Height	Float64
`nationality`	Nationality	Object

Table 1.2: Null Values in *cyclist.csv*

Attribute	**Null Values**
`_url`	0
`name`	0
`birth_year`	13
`weight`	3056
`height`	2991
`nationality`	1

Table 1.3: `.describe()` Summary for *cyclist.csv*

	`birth_year`	`weight`	`height`
Count	6121	3078	3143
Mean	1974.07	68.66	179.82
Std Dev	15.54	6.35	6.44
Min	1933	48	154
25%	1962	64	175
50%	1974	69	180
75%	1987	73	184
Max	2004	94	204

Upon initial inspection, the dataset contains no duplicate rows, as confirmed by the `.duplicated().sum()` method returning zero. However, there are significant null values in the `weight` and `height` attributes, with 3056 and 2991 missing entries respectively (Table 1.2). The `.describe()` method (Table 1.3) indicates that the numerical attributes fall within expected ranges for professional cyclists.[1]

1.1.2 Races Dataset (*races.csv*)

Similarly, the **races** table's attributes are detailed in Table 1.4.

Table 1.4: Attributes Overview for *races.csv*

Attribute Name	**Semantics**	**Type**
`_url`	Identifier for the race, includes race name, year, and stage	Object
`name`	Official name of the race	Object
`points`	Points assigned, indicating prestige/importance	Float64
`uci_points`	Points based on UCI rankings	Float64
`length`	Total distance of the race (meters)	Float64
`climb_total`	Total meters climbed (elevation difficulty)	Float64
`profile`	Race's terrain profile (flat, hilly, mountainous, etc.)	Float64
`startlist_quality`	Quality/strength of participants	Int64
`average_temperature`	Temperature during the race (Celsius)	Float64
`date`	Date of the race	Object
`position`	Final position of a specific cyclist	Int64
`cyclist`	Unique cyclist ID (_url in cyclist table)	Object
`cyclist_age`	Age of the cyclist	Float64
`is_tarmac`	Boolean: Race on tarmac	Bool
`is_cobbled`	Boolean: Race on cobbles	Bool
`is_gravel`	Boolean: Race on gravel	Bool
`cyclist_team`	Team the cyclist was part of	Object
`delta`	Time difference (seconds) from the race winner	Float64

Table 1.5: Part 1: `.describe()` Summary for *races.csv*

	`points`	`uci_points`	`length`	`climb_total`	`profile`
Count	589388	251086	589865	442820	441671
Mean	89.22	74.60	166776.18	2330.47	2.61
Std Dev	54.43	100.95	64545.61	1375.71	1.49
Min	18.00	6.00	1000.00	2.00	1.00
25%	50.00	16.00	152500.00	1309.00	1.00
50%	80.00	60.00	178200.00	2255.00	2.00
75%	100.00	100.00	203500.00	3273.00	4.00
Max	350.00	800.00	338000.00	6974.00	5.00

The races dataset contains no duplicate rows, as indicated by the `.duplicated().sum()` method returning zero. However, there are considerable null values in several attributes, with `uci_points` and `average_temperature` exhibiting over 50% missing data. The `.describe()` method results (Tables 1.5 and 1.6) reveal that most numerical attributes are within expected ranges, with minor anomalies such as a 13-year-old and a 56-year-old cyclist.

[1] More on the notebook

Table 1.6: Part 2: `.describe()` Summary for *races.csv*

	startlist_quality	average_temperature	position	cyclist_age	delta
Count	589865	29933	589865	589752	589865
Mean	1101.16	21.73	74.22	28.49	418.29
Std Dev	380.59	5.89	48.40	3.86	842.96
Min	115.00	10.00	0.00	13.00	-6906.00
25%	844.00	17.00	32.00	26.00	10.00
50%	988.00	22.00	70.00	28.00	156.00
75%	1309.00	26.00	112.00	31.00	624.00
Max	2047.00	36.00	209.00	56.00	61547.00

1.2 Initial Data Insights

Understanding the distribution and relationships within the data is crucial for subsequent analysis. Below, we present key observations derived from the datasets.

1.2.1 Key Distributions

- **Weight and Height**: both attributes for cyclists display a normal distribution. Weight clusters between 60 and 75 kg, peaking around 70 kg, while height ranges from 170 to 185 cm, peaking near 180 cm. There are no significant outliers observed.

- **Points and UCI Points**: the `points` attribute in the races dataset is heavily concentrated between 40 and 100, with a peak between 70 and 80. Values above 100 are rare, reaching a maximum of 350. Conversely, `uci_points` has over 57% missing values, and the non-null entries are skewed with most below 120, creating a long tail distribution.

- **Cyclist Age**: ages are normally distributed around 30 years, predominantly between 25 and 35 years. There are few instances of cyclists younger than 20 or older than 40, aligning with typical professional cycling demographics.

- **Delta**: this attribute, representing the time difference from the race winner, is mostly positive. However, the presence of negative values suggests potential data-entry errors. After removing the top 5% of extreme values, the distribution peaks sharply near zero, indicating most cyclists finish close to the leader.

1.2.2 Correlation Analysis

Figure 1.1 presents the correlation matrix for the **Cyclists** dataset:

- `weight` and `height` have a strong positive correlation ($\rho \approx 0.72$), aligning with physiological expectations.

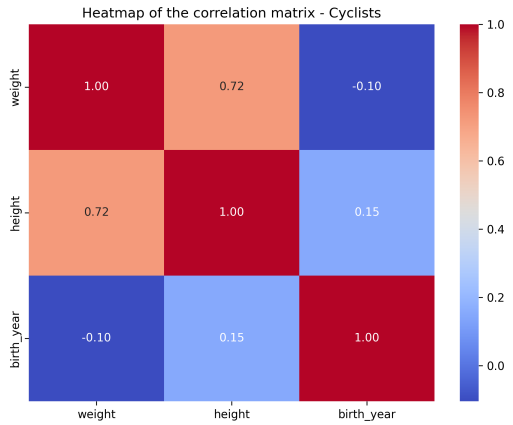

Figure 1.1: Cyclists Dataset Correlation Matrix

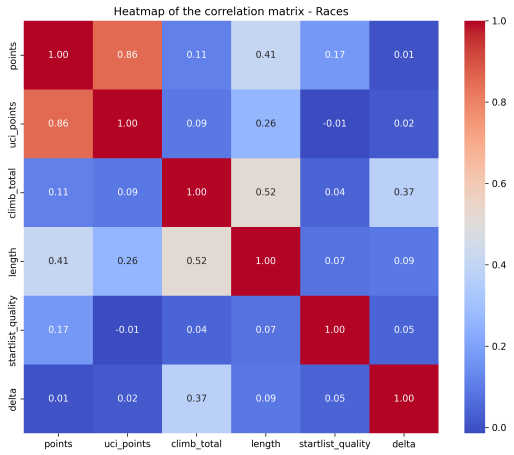

Figure 1.2: Races Dataset Correlation Matrix

- A weaker positive correlation ($\rho \approx 0.15$) exists between `height` and `birth_year`, potentially indicating that newer generations of cyclists are slightly taller on average.
- `weight` and `birth_year` show a minor negative correlation ($\rho \approx -0.10$), suggesting recent generations may weight slightly less, though the effect is minimal.

Figure 1.2 illustrates the correlation matrix for the **Races** dataset:

- `points` and `uci_points` show a strong positive correlation ($\rho \approx 0.86$), indicating they scale together.
- `length` and `climb_total` have a moderate positive correlation ($\rho \approx 0.52$), suggesting longer races typically involve more climbing.
- `points` and `length` exhibit a weaker positive correlation ($\rho \approx 0.41$), implying that longer races might offer more points.
- Attributes like `startlist_quality` and `delta` show very weak correlations with other variables, indicating minimal linear relationships.

Chapter 2

Data Transformation

2.1 Handling Missing Values

Managing missing values was essential for preparing the data. In the cyclist dataset, nearly half of the **weight** and **height** values were missing. While we initially planned to infer one attribute from the other when only one was missing, this applied to just 79 out of 2984 cases. Thus, we opted for a median imputation to preserve data distribution.

Missing values in **birth_year** were computed from **cyclist_age** in the races dataset. Rows missing both were filled by imputing **cyclist_age** with the median and calculating the corresponding **birth_year**. A double-check ensured consistency between **cyclist_age** and **birth_year**, leveraging **name** and **_url** for verification. Minor discrepancies due to typos or middle names were deemed acceptable as URLs uniquely identified cyclists.

For **nationality**, a single null value was manually corrected to align with the races dataset. Post-imputation, distributions of **weight** and **height** shifted but remained suitable for analysis.

2.2 Type Casting

To ensure consistency, we performed the following type conversions:

- **birth_year**, **profile**, **climb_total**, **cyclist_age**, and **points** were converted from float to integer.
- **date** was converted from object to a proper date format.

These adjustments improved data integrity and prepared the datasets for further analysis.

2.3 Column Removal

To simplify the dataset and focus on relevant features, we decided to drop several columns:

- **average_temperature**: this column contained too many missing values, and imputing them was deemed impractical due to the complexity of accurately estimating temperature data.
- **is_cobbled** and **is_gravel**: these columns were removed because the totality of their values were **False**, leaving only the boolean **is_tarmac** to represent road surface types effectively.
- **uci_points**: with more than half of its values missing, this column was deemed redundant. Through checks, we confirmed that stages either entirely lacked or fully contained **uci_points**, without partial entries. Since the **points** column already provides similar information with consistent population, **uci_points** was considered unnecessary and removed.

2.4 Feature Engineering and Novel Feature Definition

To enhance the dataset and support advanced analyses, we introduced several new features by deriving meaningful information from existing attributes:

- **BMI (Body Mass Index)**: added to the cyclists dataset, this feature is calculated as the weight (in kilograms) divided by the square of the height (in meters). BMI provides a standardized measure of body composition and complements the physical attributes of cyclists.
- **continent**: derived by mapping each cyclist's nationality to their corresponding continent, this feature adds a geographical dimension to the analysis.
- **avg_position**: introduced for further computations, particularly in clustering tasks, this feature captures the average performance position of cyclists.
- **avg_delta**: this feature, which measures average time gaps, will be computed later after cleaning the **delta** column to remove outliers.
- **season**: to categorize the time of year when races occurred, we added a feature representing the four seasons, enhancing temporal analysis of race events.
- **race_intensity**: this feature represents the overall difficulty of a race, computed as a weighted combination of:
    1. The ratio of **climb_total** to **length**, indicating race steepness.

2. The **profile** score, reflecting the terrain's difficulty.
3. A logarithmic transformation of **length**, capturing the influence of race distance on intensity.

2.5 Outlier detection

After introducing the new features, we proceeded with outlier detection to identify and manage anomalous data points. Two principal methods were used: the **Z-score** for approximately normal distributions and the **Interquartile Range (IQR)** method for non-normal distributions.

- **birth_year**: the IQR method was employed due to its non-normal distribution. No anomalies were detected.
- **weight**, **height**, and **bmi**: using the Z-score method, we found that the few flagged values for **weight** and **height** were plausible for professional cyclists. For **bmi**, some values below 18 (underweight) or above 25 (overweight) emerged. However, these values were retained as they are realistic for athletes.
- **points**: no method was applied due to the varied and unrelated ranges of values, making outlier detection less meaningful.
- **length** and **climb_total**: the IQR method was applied due to their asymmetric and multimodal distributions. Although several points were flagged, they were deemed plausible and retained.
- **cyclist_age**: both the Z-score and IQR methods were applied. While the IQR method flagged over 4,000 entries, the Z-score identified only one outlier. A deeper analysis revealed two cyclists with unusual ages (13 and 56 years), which were retained as plausible.
- **delta**: negative values were replaced with the preceding row's value to ensure consistent progression of delays with increasing position. The IQR method confirmed no further anomalies.
- **race_intensity**: using the IQR method, no outliers were detected, with all values falling within a reasonable range.

Chapter 3

Clustering Analysis

In this chapter, we present a detailed examination of clustering methods applied to five pairs of variables from our cyclist dataset. Our objective is to uncover hidden groupings that reveal structural and performance differences among riders. Below, we describe the key steps involved in each clustering approach, along with the main numerical and qualitative observations.

1. *Weight* and *Height*
2. *BMI* and *Average Position*
3. *Average Delta* and *Average Position*
4. *Points* and *Race Intensity*
5. *Points* and *Startlist Quality*

For each subset, duplicates related to `data_pri` and `data_psq` were removed to ensure that each slice focuses on the characteristics of interest, avoiding repeated races or entries differing solely by participant composition. This preprocessing step was critical for ensuring the validity and clarity of our clustering results, particularly given the potential for biases introduced by redundant data.

Rationale Behind Feature Pairs

Each feature pair was chosen to capture a distinct facet of a cyclist's profile, aiming to balance physiological, competitive, and environmental factors:

- ***Weight* and *Height***: highlights physical stature and its clustering implications, such as whether taller and heavier cyclists exhibit grouping tendencies based on morphological similarities.
- ***BMI* and *Average Position***: examines the relationship between body composition and competition outcomes, providing insight into how body metrics influence race success.

- ***Average Delta* and *Average Position***: analyzes the impact of pacing consistency on performance, revealing patterns among riders with varying strategies and stability.
- ***Points* and *Race Intensity***: explores the link between success and race demands, uncovering clusters based on athletes' adaptability to high-intensity environments.
- ***Points* and *Startlist Quality***: investigates performance trends in varying levels of competition, distinguishing consistent elite performers from situational high achievers.

3.1 Normalization Techniques

To harmonize attributes with varying scales, two normalization strategies were employed. These techniques ensured that no single feature dominated the clustering process:

- **StandardScaler**: centers features around zero with unit variance, retaining outliers but reducing their influence. This approach often yielded stable and interpretable clusters, particularly when outlier effects were meaningful for distinguishing groups.
- **Min-Max Scaler**: rescales values to the [0, 1] range, enhancing compactness but occasionally compressing extreme values. While this method was advantageous for visualizing data distributions, it sometimes led to less distinct clusters for broad-range features.

In practice, **StandardScaler** often yielded more stable clusters. **Min-Max Scaler** was useful for highlighting extreme values but risked excessive compression, making it ideal for datasets with smaller variability ranges.

3.2 K-Means Clustering

K-Means partitions data into generally spherical clusters. The **Elbow method** indicated that three clusters captured the main patterns, with occasional benefits from a fourth cluster. This adaptability made K-Means a suitable choice for exploring broad population trends while accommodating finer granularity when needed.

Qualitative Observations

- ***Weight* and *Height***: naturally grouped into "shorter and lighter," "medium," and "taller and heavier." The absence of extreme outliers reflected the professional norms of the dataset, where athletes exhibit consistent physical training standards.

- ***BMI* and *Average Position***: diverse body compositions in well-trained riders highlighted that BMI alone does not dictate success. Clustering often separated groups based on nuanced performance traits rather than simplistic physical metrics. As we can see from 3.1 we have here four clasters, this is only true for **StandardScaler**.

- ***Average Delta* and *Average Position***: StandardScaler highlighted uniform pacing, while Min-Max emphasized segmentation into smaller groups. This duality underscores the role of consistent pacing as a factor influencing competitive stability.

- ***Points* and *Race Intensity***: clusters of mid-range performers contrasted with high-intensity specialists, demonstrating how adaptability to race demands correlates with competitive outcomes.

- ***Points* and *Startlist Quality***: clustering revealed three distinct groups of cyclists based on their points and startlist quality. Both standardized and Min-Max scaled data resulted in well-separated clusters, with Min-Max scaling producing tighter and more cohesive groupings.

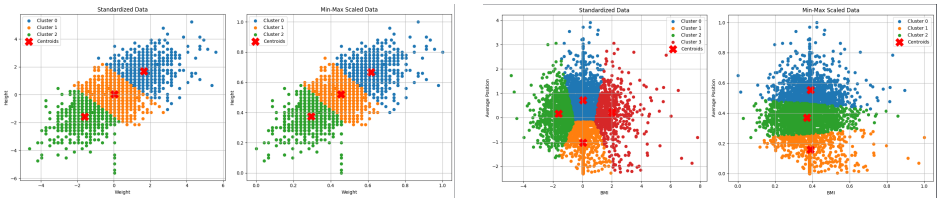

Figure 3.1: K-Means Clustering results for Weight vs Height and BMI vs Average Position.

3.3 DBSCAN

DBSCAN identifies clusters of arbitrary shapes and isolates noise points based on density thresholds (ϵ and *min samples*). ϵ represent the maximal distance between 2 points belonging to the same cluster, and *min samples* the minimum number of points to define a cluster. We always used *min sample* $= 3$. For what concerns ϵ we use the elbow method using K-NN distance graph. We try to estimate the most suitable epsilon value, in case the estimate is not satisfactory we proceed to manually modify this value. This method was particularly effective in identifying outliers and clusters with irregular geometries.

Silhouette Score is a metric that measures how well each data point fits within its assigned cluster compared to other clusters. It ranges from -1 to +1. **Cohesion** refers to the degree of similarity or closeness among the data points within the same cluster.

Qualitative Observations

- ***Weight* and *Height***: mainstream clusters with side groups for athletes, whose extreme attributes (e.g., height, muscle mass) naturally set them apart. Noise points often represented unique cases not fitting general patterns.
- ***BMI* and *Average Position***: most athletes clustered around average BMI values, while smaller groups captured specific performance trends. Outliers often included riders whose race results diverged significantly from their physical profiles.
- ***Average Delta* and *Average Position***: dominant cluster with smaller groups for specialists. Noise points highlighted inconsistent racers or atypical performance variations.
- ***Points* and *Race Intensity***: in this case, there are a number of clusters that are well separated from each other for the Standardized data, while for the data treated with Min-Max there are more clusters that are still well spaced from each other.
- ***Points* and *Startlist Quality***: the values have highlighted a practically identical division of the dataframe which consists of 6 clusters without noise values, for obvious reasons also silhouette score and cohesion are equal.

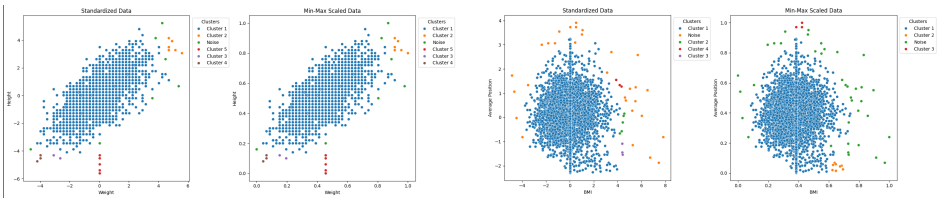

Figure 3.2: DBSCAN results for Weight vs Height and BMI vs Average Position. Noise points are highlighted, offering insights into unique cases.

3.4 Hierarchical Clustering

Hierarchical clustering explores data at varying levels of granularity by iteratively merging or splitting clusters.

- **Linkage Methods**: Different linkage methods significantly affect the clustering results. Ward linkage consistently produced compact and interpretable clusters by minimizing the total within-cluster variance. Complete linkage balanced cluster cohesion with moderate sensitivity to outliers, while single linkage often resulted in chaining effects, linking distant points into elongated structures. Similarly, Divisive Hierarchical Clustering, which starts with a single

cluster and recursively splits it, complements agglomerative methods by revealing distinct groupings that may not be apparent through merging alone.

- **Cluster Granularity**: By analyzing dendrograms, we observed that larger clusters could be split into smaller, more refined subgroups. Divisive Clustering further enhanced this granularity by effectively identifying subgroups within the main clusters, providing a more detailed understanding of athletes performance and characteristics.

- **Outliers**: Single linkage tended to incorporate extreme values into elongated clusters, potentially obscuring meaningful distinctions. Ward linkage, in contrast, effectively isolated these outliers, forming distinct small clusters. For example, outliers in *BMI* vs *Average Position* represented cyclists with atypical body compositions achieving rare race outcomes. Divisive Clustering complemented this by systematically splitting clusters to isolate outliers, ensuring that extreme values were appropriately categorized without distorting the overall cluster structure.

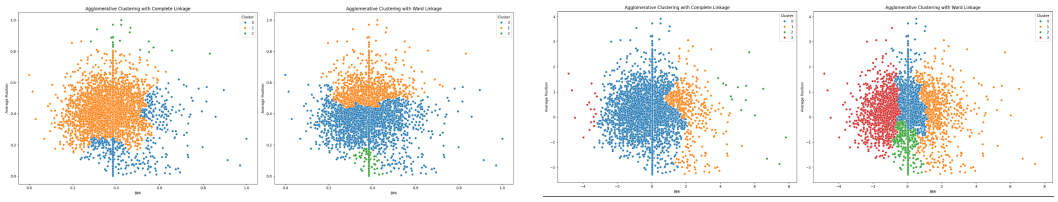

Figure 3.3: Complete and Ward linkage with Min-Max and Standardized data for Avg position and BMI.

From the images provided, the hierarchical clustering analysis reveals clear separations. Using Min-Max scaling, the clustering appears to exaggerate distinctions between smaller subgroups, particularly in *Weight* vs *Height*, where athletes formed compact clusters. On the other hand, StandardScaler normalization generated more balanced clusters, facilitating the identification of broader groupings.

The dendrogram analysis highlighted that in cases such as *Average Delta* and *Average Position*, increasing the threshold for linkage distance effectively separated clusters of high-consistency riders from those with more variable performances.

Silhouette scores for hierarchical clustering varied significantly depending on the linkage method:

- Ward linkage typically yielded scores between 0.42 and 0.58, indicating moderately defined clusters.

- Complete linkage performed better in datasets with less overlap, scoring up to 0.62 in *Points* and *Startlist Quality*.

- Single linkage often resulted in lower scores (0.30–0.45), reflecting its sensitivity to chaining.

3.5 OPTICS Clustering

OPTICS (Ordering Points To Identify the Clustering Structure) is a density-based clustering algorithm designed to identify clusters of varying densities within a dataset. Unlike algorithms such as DBSCAN, OPTICS does not require a predefined number of clusters and is particularly effective at detecting clusters with different shapes and densities, making it more flexible for complex datasets.

The algorithm operates by calculating, for each point, the *core distance*, which is the minimum distance required to consider that point as a core point based on a specified radius parameter. Subsequently, the *reachability distance* is determined, representing the distance needed to reach a point from another core point. This process orders all points in the dataset based on their reachability distance, creating a structure that facilitates visualization and identification of clusters through a reachability plot. Clusters are identified as valleys in the plot, while noise points and outliers appear as peaks with high reachability distances.

Clustering Results by Feature Pairs

- ***Weight* and *Height***: applying OPTICS with *radius* = 0.4 (standardized) and 0.04 (Min-Max) and *neighbors* = 5, a primary cluster consisting of 6,063 and 6,068 cyclists was identified for standardized and Min-Max scaled data, respectively, with 71 and 66 noise points. The resulting plots show a similar clustering structure for both normalization methods, highlighting a large main cluster while excluding outer values as noise. The consistency between standardized and Min-Max scaled data suggests that weight and height are strongly correlated features, leading to the formation of a dominant cluster. The relatively low number of noise points indicates that most cyclists fall within this main grouping, with only a few outliers differing significantly in weight or height.

- ***BMI* and *Average Position***: using the same parameters (*radius* = 0.4, 0.04; *neighbors* = 5), the analysis revealed a primary cluster with 6,075 cyclists and a small secondary cluster with 6 cyclists in the standardized data, and two small clusters with 4 and 2 cyclists in the Min-Max scaled data. Silhouette scores were 0.550 for standardized data and 0.242 for Min-Max scaled data, indicating moderate separability of clusters, particularly better in the standardized normalization. The higher silhouette score for standardized data indicates that the clusters are more distinct and better separated when the data is standardized, enhancing the algorithm's ability to differentiate between the main group and outliers.

- ***Average Delta* and *Average Position***: applying OPTICS with *radius* = 0.4 (standardized) and 0.04 (Min-Max) and *neighbors* = 5 resulted in a single large cluster containing 6,105 cyclists in the standardized data and 6,071 cyclists in the Min-Max scaled data, accompanied by 29 and 44 noise points, respectively.

Additionally, two small clusters with 10 and 9 cyclists were identified in the Min-Max scaled data. The silhouette score was intermediate (0.384) for the Min-Max scaled data, while it was not defined for the standardized data. The dominant cluster in both normalization methods indicates a general trend in average delta and position among most cyclists. The emergence of small clusters in the Min-Max scaled data may point to specific performance patterns that are less pronounced or absent in the standardized data. The intermediate silhouette score for Min-Max scaled data suggests a reasonable, though not strong, separation between the main cluster and the smaller groups.

- ***Points* and *Race Intensity***: using *radius* = 0.5 (standardized) and 0.1 (Min-Max) with *neighbors* = 5, the algorithm identified a primary cluster with 4,713 cyclists in both standardized and Min-Max scaled data. In the standardized data, six additional secondary clusters were detected without any noise points, whereas the Min-Max scaled data revealed ten additional clusters with 7 noise points. Silhouette scores were 0.173 for standardized data and 0.006 for Min-Max scaled data, indicating poor separability in the Min-Max scaled data. The significantly lower silhouette scores, especially for Min-Max scaled data, indicate that the multiple clusters overlap considerably, making them less distinct. The standardized data's ability to identify secondary clusters without introducing noise points reflects a better separation of distinct cyclist profiles based on their performance metrics.

- ***Points* and *Startlist Quality***: with *radius* = 0.4 (standardized) and 0.04 (Min-Max) and *neighbors* = 5, OPTICS identified well-defined clusters in both normalization methods. In the standardized data, a primary cluster of 4,670 cyclists was supported by five smaller clusters and a single noise point, with a silhouette score of 0.205. In the Min-Max scaled data, twelve clusters were identified with 23 noise points and a silhouette score of 0.059, indicating less optimal separability compared to the standardized data. The standardized data reveals a clear main group of cyclists with high startlist quality and points, along with a few distinct subgroups, suggesting specific performance tiers or roles within races. The Min-Max scaled data's increased number of clusters and higher noise points indicate a fragmented clustering structure, likely due to the scaling method making it harder to distinguish meaningful groupings. The higher silhouette score for standardized data underscores its effectiveness in producing more coherent and interpretable clusters.

3.6 CURE Clustering

CURE (Clustering Using Representatives) is a hierarchical clustering algorithm optimized for large datasets and diverse cluster shapes. It begins by randomly sampling the dataset to reduce computational complexity, treating each sampled point as an

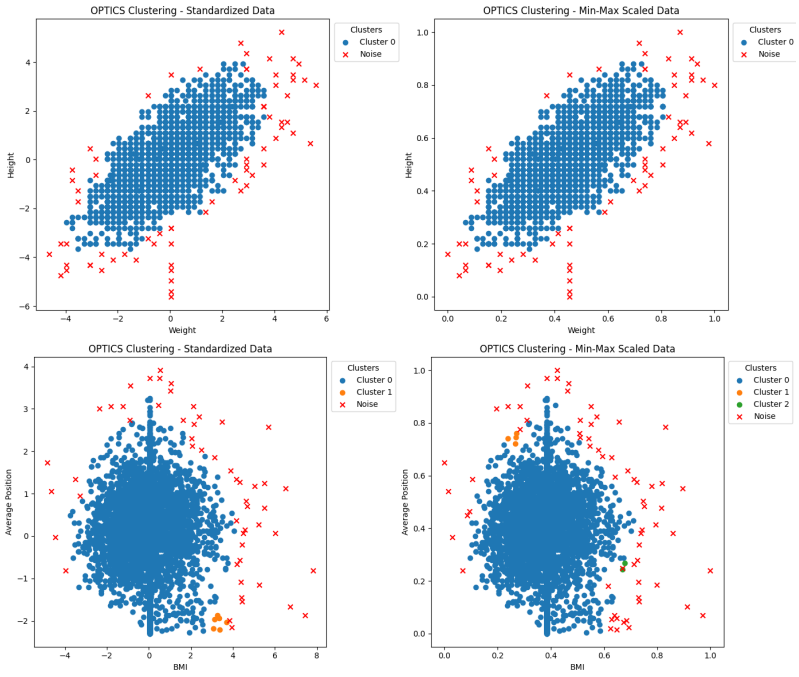

Figure 3.4: OPTICS Clustering Results for both Standardized and Min-Max: Weight vs. Height. BMI vs. Average Position.

individual cluster. The algorithm iteratively merges clusters based on the closest pair of representative points. For each cluster, CURE selects a fixed number of well-scattered *representative points* and shrinks them toward the cluster centroid to minimize outlier influence. This method balances compactness and flexibility, enabling CURE to effectively identify non-spherical and irregular clusters, surpassing traditional methods like K-Means.

Clustering Results by Feature Pairs

- ***Weight* and *Height***: applying CURE with three clusters resulted in a dominant cluster under both normalization methods. Standardized data emphasized a primary cluster with 6,110 cyclists, while Min-Max scaling highlighted a slightly larger main cluster with 6,120 cyclists. The silhouette scores were 0.655 for standardized data and 0.720 for Min-Max scaling, indicating well-defined clusters, especially with Min-Max scaling.

- ***BMI* and *Average Position***: for this feature pair, both standardized and Min-Max scaled data identified a predominant cluster comprising 6,130 cyclists. The silhouette scores were 0.720 for standardized data and 0.607 for Min-Max scaling, reflecting consistent clustering with minimal smaller clusters across both normalization methods.

- ***Average Delta* and *Average Position***: applying CURE revealed that stan-

dardized data produced a main cluster with 6,118 cyclists, achieving a silhouette score of 0.630. Min-Max scaling resulted in a similar primary cluster with 6,124 cyclists and a slightly lower silhouette score of 0.622. Both normalization techniques effectively separated the main cluster from the other two smaller clusters.

- ***Points* and *Race Intensity***: in this analysis, standardized data yielded a prominent cluster with 4,771 cyclists (Silhouette Score: 0.590), whereas Min-Max scaling produced a more dispersed distribution with a primary cluster containing 3,914 cyclists (Silhouette Score: 0.550). Standardization provided a clearer separation of clusters compared to Min-Max scaling.

- ***Points* and *Startlist Quality***: applying CURE identified three well-formed clusters under both normalization methods. Standardized data featured a primary cluster with 4,771 cyclists (Silhouette Score: 0.611), while Min-Max scaling resulted in an equally sized primary cluster with 4,771 cyclists but with a slightly lower silhouette score of 0.596. Both methods effectively distinguished cyclists based on their points and startlist quality, with Min-Max scaling offering marginally better cluster compactness.

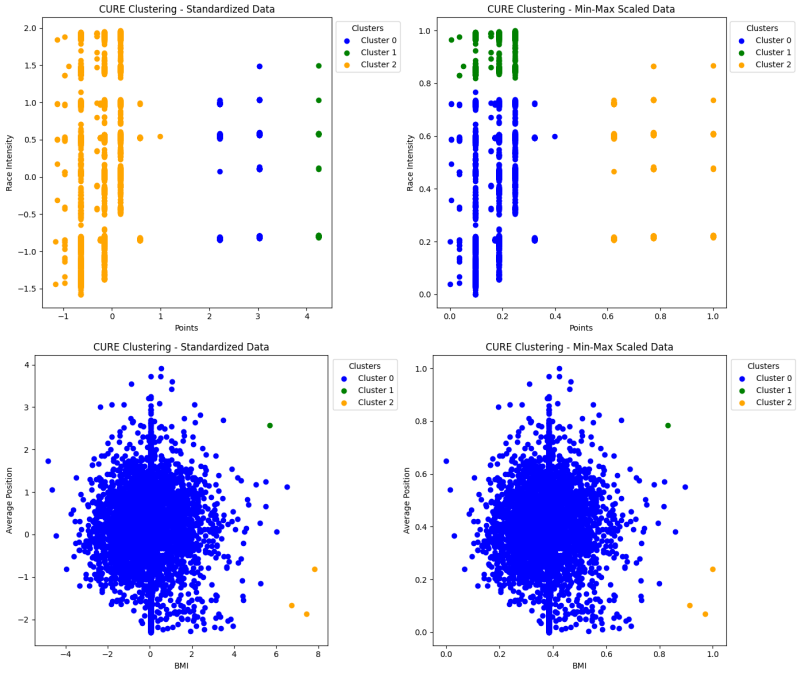

Figure 3.5: CURE Clustering Results for both Standardized and Min-Max: Points vs. Race Intensity. BMI vs. Average Position.

3.7 General Observations

- **Normalization Impact**: Both Min-Max scaling and standardization improved the performance of clustering methods, with standardization being more consistent across feature pairs.
- **Method Performance**: Ward Linkage and K-means proved robust across most feature pairs, while Single Linkage struggled to handle noise and outliers effectively.
- **Noise and Outliers**: DBSCAN and OPTICS showed potential for handling noise, with better results under standardized data.

Chapter 4

Prediction

4.1 Data Preparation

Before employing the learning algorithms for our prediction purpose, we first need to rearrange our data. To begin, we merge `cyclists` and `races` into `data_merged` by linking the `_url` column from `cyclists` and the `cyclist` column from `races`. Performing the merge, some mismatched entries come out: we cancel them because they concern a cyclist who did not participate in any race.

Then, we delete[1] the following useless columns: `name` column coming from `cyclist` dataset, `name` column coming from `races` dataset, `weight`, `height`, `avg_position`, `avg_delta`, `position`, `birth_year`, `cyclist_team`; `date` has been modified to contain only the year.

Subsequently, we add the binary column `top_20` to express if a certain cyclist is in the top-20 of a certain race (=1) or not (=0). Afterwards, we employ the `discretize_data` method to transform the categorical columns (`race_url`, `season`, `cyclist_url`, `nationality`, `continent`) into numerical ones. The only other casting we perform is transform the `is_tarmac` boolean column. To end, we prepare the dataset for the learning algorithms: `train_data` contains the dataset regarding the races before 2022, `test_data` contains the dataset regarding the races from 2022. Moreover, these two has been divided in:

- `train_feature`, all attributes except the target one (`top_20`) for **training**
- `train_target`, the target we want to predict (`top_20`) for **training**
- `test_feature`, all attributes except the target one (`top_20`) for **testing**
- `test_target`, the target we want to predict (`top_20`) for **testing**

[1] More on the notebook

4.2 Learning Algorithms

The models included are: Decision Tree, Random Forest, AdaBoost, Naïve Bayes, K-Nearest Neighbors, and Neural Network. We tried to include XGBoost, Rule-based, and SVM, but their prolonged compilation times prevented completion.

For every model, we show its performance using the `report_scores` method, the Confusion Matrix, and a corresponding plot when feasible.

4.2.1 Tree-Based Models

Beginning with the **Decision Tree Model**, the obtained tree is shown below:

Figure 4.1: Decision Tree

The first feature considered for the division is `delta`, then `climb_total` and `race_intensity`. The coloring of the nodes represents the two classes of the problem, orange for 1 (= Top 20) and blue for 0 (= Non-Top 20), while its intensity represents how sure the prediction is (the more the color is intense, the more the prediction is valid).
The performance and the confusion matrix of the Decision-Tree and **Random Forest** models are reported below:

	Precision	Recall	F1 score	Support
Non-Top 20	0.95	0.75	0.84	30466
Top 20	0.33	0.77	0.46	4940
Accuracy			0.75	35406
Macro avg	0.64	0.76	0.65	35406
Weighted avg	0.87	0.75	0.79	35406

Table 4.1: DT Performance

	Precision	Recall	F1 score	Support
Non-Top 20	0.96	0.81	0.88	30466
Top 20	0.40	0.79	0.53	4940
Accuracy			0.81	35406
Macro avg	0.68	0.80	0.71	35406
Weighted avg	0.88	0.81	0.83	35406

Table 4.2: RF Performance

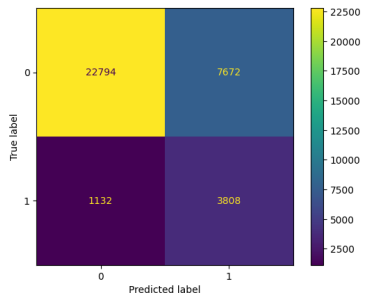

Figure 4.2: DT Confusion Matrix

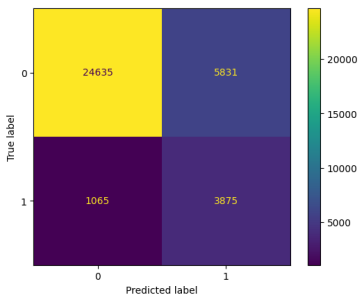

Figure 4.3: RF Confusion Matrix

4.2.2 AdaBoost

We implemented two versions of the AdaBoost model: the first one considers its default configurations, while the second employs the previously computed Decision Tree as a base.

	Precision	Recall	F1 score	Support
Non-Top 20	0.87	1.00	0.93	30466
Top 20	0.85	0.05	0.10	4940
Accuracy			0.87	35406
Macro avg	0.86	0.53	0.51	35406
Weighted avg	0.86	0.87	0.81	35406

Table 4.3: AdaBoost Performance

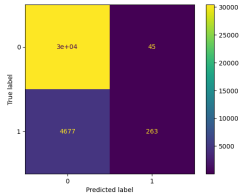

Table 4.4: AB Conf. M.

This model has difficulties to recognize the Top 20 values. In the Tree-Based Models, we addressed the issue introducing the parameter `class_weight = 'balanced'` to higher the weight of the minor class. In the AdaBoost, we can improve the results modifying the parameters, employing 200 estimators and a lower learning rate (= 0.1); this version better recognizes the True Positive, paying on the other hand a higher number of False Positive.

	Precision	Recall	F1 score	Support
Non-Top 20	0.96	0.75	0.84	30466
Top 20	0.34	0.80	0.47	4940
Accuracy			0.75	35406
Macro avg	0.65	0.77	0.66	35406
Weighted avg	0.87	0.75	0.79	35406

Table 4.5: EAB Performance

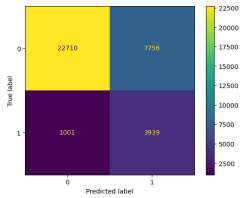

Table 4.6: EAB Conf. M.

4.2.3 Naïve Bayes

The performance and the confusion matrix of the Naïve Bayes are shown below.

	Precision	Recall	F1 score	Support
Non-Top 20	0.88	0.88	0.88	30466
Top 20	0.24	0.23	0.23	4940
Accuracy			0.79	35406
Macro avg	0.56	0.56	0.56	35406
Weighted avg	0.79	0.79	0.79	35406

Table 4.7: Naïve Bayes Performance

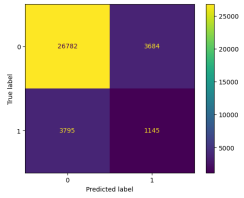

Table 4.8: NB Conf. M.

As highlighted for AdaBoost, there is not a way to directly balance the weights, so the model finds it difficult to find the true 1 values (the Top 20). Anyway, it works better than AdaBoost basic version.

4.2.4 K-Nearest Neighbors

	Precision	Recall	F1 score	Support
Non-Top 20	0.90	0.86	0.88	30466
Top 20	0.30	0.38	0.34	4940
Accuracy			0.79	35406
Macro avg	0.60	0.62	0.61	35406
Weighted avg	0.81	0.79	0.80	35406

Table 4.9: KNN Performance

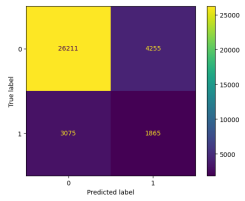

Table 4.10: KNN Conf. M.

As highlighted for previous two models, there is not a way to directly balance the weights, so the model finds it difficult to find the True Positive values (the Top 20). Anyway, it works better than AdaBoost.

4.2.5 Neural Network

	Precision	Recall	F1 score	Support
Non-Top 20	0.87	1.00	0.93	30466
Top 20	0.76	0.06	0.11	4940
Accuracy			0.87	35406
Macro avg	0.81	0.53	0.52	35406
Weighted avg	0.85	0.87	0.81	35406

Table 4.11: NN Performance

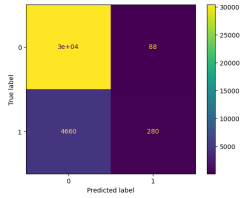

Table 4.12: NN Conf. M.

As we can see, this model is one of the worst among all for identifying the Top-20 values, but recognizes almost perfectly the Non Top-20 values (as standard AdaBoost). To conclude this chapter, we present a comparison of the models' accuracy:

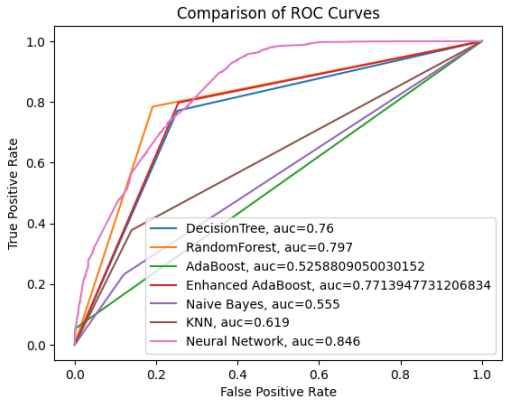

Figure 4.4: Comparison of model accuracies

Chapter 5

Explanation

To deal with this task, we imported from the previous some of the models we implemented, in particular we focus on Decision Tree and Random Forest. We also experimented with the use of a Neural Network, which obtained the best accuracy value among the models tested.

However, the implementation encountered a number of difficulties, including incompatibilities between libraries, kernel-blocking methods, high execution times and issues related to the configuration of the development environment, particularly with Visual Studio Code. The experiments conducted have been documented and reported at the end of this notebook as part of this Task. Here we focus on feature importance, rule explanation and counterfactual instances for the two using the **SHAP**, **LIME** and **LORE** libraries.

For the feature importance subtask, a utility function `plot_feature_importance` has been written to detect the most influent attributes for building the model and to plot them also.

5.1 Decision Tree

The most relevant feature is `delta`: it is meaningful since the lower this value is, the closer the cyclist is to first place (and therefore to the Top-20). Other influent attributes are: `race_intensity`, `climb_total`, `date`, `bmi`, `startlist_quality`, `race_url` (Figure 5.1).

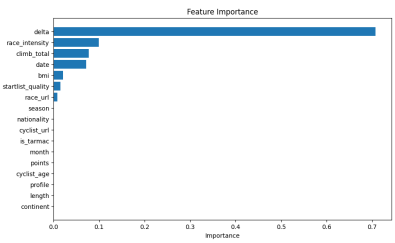

Figure 5.1: DT Feature Importance

Then, it can be affirmed that personal origin and age don't influence performances (as the majority of the cyclists come from 5 countries which is really few and age range isn't so wide), but only physical form does (see `bmi`); moreover, when the race is taken makes no importance (also because the majority are taken during spring/summer).

All features taken into account by the model, except from `bmi`, are races-specific. We can confirm what we said thanks to SHAP library and its two approaches, interventional and distributional, regarding the 1 class (= Top-20) as we can see in Figure 5.2 and 5.3.

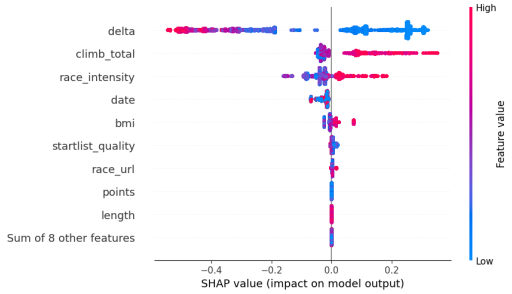

Figure 5.2: DT Interventional SHAP

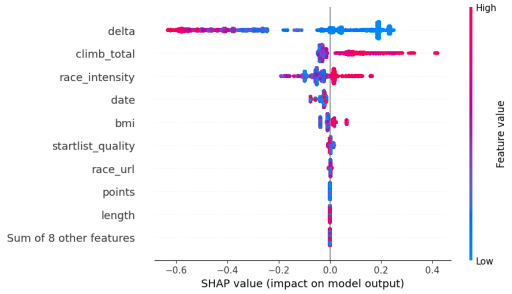

Figure 5.3: DT Distributional SHAP

To sum up, they are very similar but show how some features aren't taken into account (with all values concentrated into the 0) while the already mentioned ones are. In order to understand how the results, and thus the dependency between features, vary, we calculate the difference between the SHAP approaches for each instance and each feature. Using a kernel density estimate (KDE), the probability distribution of the calculated maximum differences is estimated (Figure 5.4).

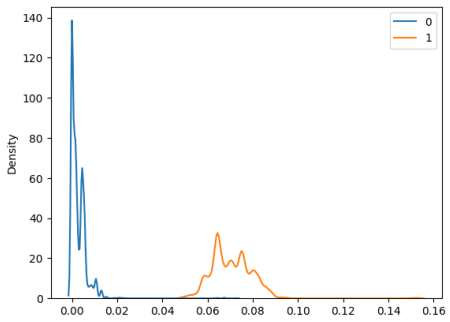

Figure 5.4: DT KDE

Instances belonging to class 0 (Non Top-20) exhibit nearly independent SHAP values between features, as the density is concentrated close to zero. Instances in class 1 (Top 20) show more dependence between the features, suggesting that the behaviour of the model to predict a positive outcome is more complexly dependent on the interactions between the variables. In the notebook can be found the impact of the difference between the two SHAP approaches at the single feature level. We proceed with **LIME** to explain the first 100 instances of the training dataset and

analyze how small perturbations of the input features affect the model predictions. The metric chosen is the f1-score, since combines precision and recall. The results we obtain for different noise levels can be classified in:

- Low noise (0.1-0.5): the performance of the model remains stable, since the f1-score does not vary, indicating that the model is robust to small perturbations.
- High noise (0.6-0.9): a slight improvement in performance is observed (f1-score increases marginally). This phenomenon is particularly evident in the `delta` feature, suggesting that high noise can, in some cases, help the model improve predictions.

For what concerns **LORE** algorithm, the explanation indicates that in order to classify the instance as `top_20` (predicted class = 1), the feature `delta` must be <= 27.5. In addition, the counterfactual rules, which highlight minimal changes required to alter the model prediction, show that in order to go from class 1 to class 0, the following conditions must be met:

- `delta` > 27.5
- `climb_total` <= 3032.5
- `race_intensity` must be simultaneously: <= 2.41 and 0.95.
- `startlist_quality` > 900.5
- `length` > 15000.0

Finally, the DICE framework is used for counterfactual analysis and shown in the notebook as a DataFrame: we only report here that changes occur mainly in the `delta` column.

5.2 Random Forest

Differently from Decision Tree, Random Forest employs more attributes even if with less influence. The most relevant feature is `delta` too, followed by all the others. The most relevant ones following `delta` are the same of the Decision Tree too, but it can be seen that also `length`, `profile` and `points` acquire importance between them (Figure 5.5) As before, it can be affirmed that personal origin, age and the period don't influence performances.

Analyzing the SHAP plots (Figure 5.6 and 5.7), in the interventional graph, the SHAP values associated with `delta` show a strong dispersion, with both very positive and very negative values distributed over a wide range. `race_intensity` and `climb_total` show some variability in SHAP values, but the dispersion is less pronounced than for delta. However, the patterns are similar: low values (blue) tend to influence the pattern in the opposite way to high values (red). `startlist_quality`

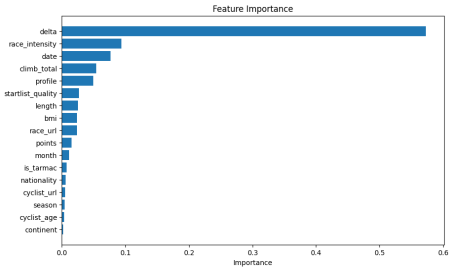

Figure 5.5: RF Feature Importance

and `points` have a narrower distribution than SHAP values, indicating that their isolated impact is smaller. For what concerns the distributional, SHAP values of `delta` show less dispersion. This suggests that the model interprets the importance of `delta` in relation to the other variables. `race_intensity` and `climb_total` show a more symmetrical and compact distribution, which reflects the preservation of dependencies between the features, as well as `startlist_quality` and `points`.

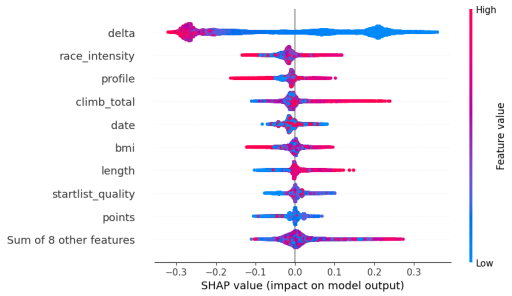

Figure 5.6: RF Interventional SHAP

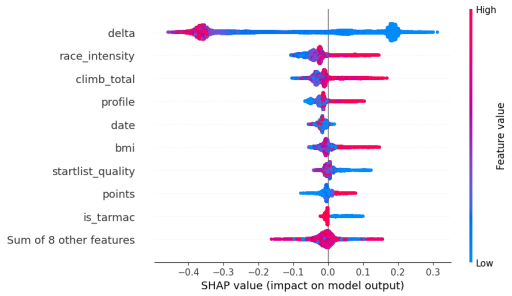

Figure 5.7: DT Distributional SHAP

As before, we compute the max differences of SHAP values between the two and compute the KDE (Figure 5.8); also here, class 0 has peaks concentrated near zero while class 1 is still concentrated around higher values. In the notebook can be found the impact of the difference between the two SHAP approaches at the single feature level.

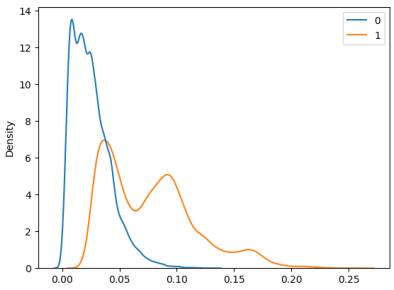

Figure 5.8: RF KDE

Also for Random Forest, **LIME** has been applied to explain the first 100 instances of the training dataset using the f1-score metrics, since combines precision and recall.

The difference between performance and `classification_report` results is mainly zero. For `delta`, there is little change in performance as the noise level increases. The observed differences are almost negligible, suggesting that the behaviour of the model is very robust to perturbations introduced by noise. This suggests that the model does not rely heavily on small variations in many of the features to determine the final result except for `delta`, which therefore has a greater impact on the model's decision-making process.

For what concerns **LORE** algorithm, the explanation indicates that in order to classify the instance as `top_20`:

- `delta` $<=$ 27.5
- `bmi` $>$ 21.30
- `cyclist_age` $<=$ 22.5

For the counterfactual rules, in order to go from class 1 to class 0, the following conditions must be met:

- `delta` $>$ 287.0
- `profile` $>$ 0

To end, the DICE framework is used for counterfactual analysis and shown in the notebook as a DataFrame. The final comparisons confirm what was shown in task 1 and task 3 confirms what was then shown by the model, i.e. no unexpected or nonsensical patterns or new ones are present.