Datamining Project: Cycling

Abdolrahim Tooranian*

January 5, 2025

1 Data Understanding

The dataset exhibits several inconsistencies and issues that need addressing. For instance, we observe negative values for `delta`, which are nonsensical. Additionally, `delta` values are sometimes even not aligned with positional ranks, where better positions must correspond to smaller `delta` values. Certain columns, such as `is_cobbled` and `is_gravel`, have uniform values across all rows. Similarly, columns such as `average_temperature` and `uci_points` have very high missing value rates even 90%. There are many other columns with missing values too, but they are more manageable.

The first notable insight can be observed in Figure 1. We see that the number of races increases over time; however, there is a sharp drop in 2020, clearly due to the impact of COVID-19.

Analyzing `delta`, we find that 20% of its values are zeros. Interestingly, the average number of racers per race is much higher, indicating that multiple zero `delta` values exist within the same race (see Figure 2). This suggests that the `delta` column contains noise and warrants further investigation.

Correlation Analysis

To examine the relationships between features, we analyzed the Spearman correlation coefficients (which captures monotonic correlation instead of only linear). These correlations, visualized in Figure 3. Normally, a high correlation exists between `height` and `weight`, which will be addressed later. A similar pattern is observed between `profile` and `climb`.

Temporal Data Quality

Apparently, data quality drops when examining older records. Columns such as `profile` and `uci_points` exhibit more missing or faulty data for earlier

*a.tooranian@studenti.unipi.it

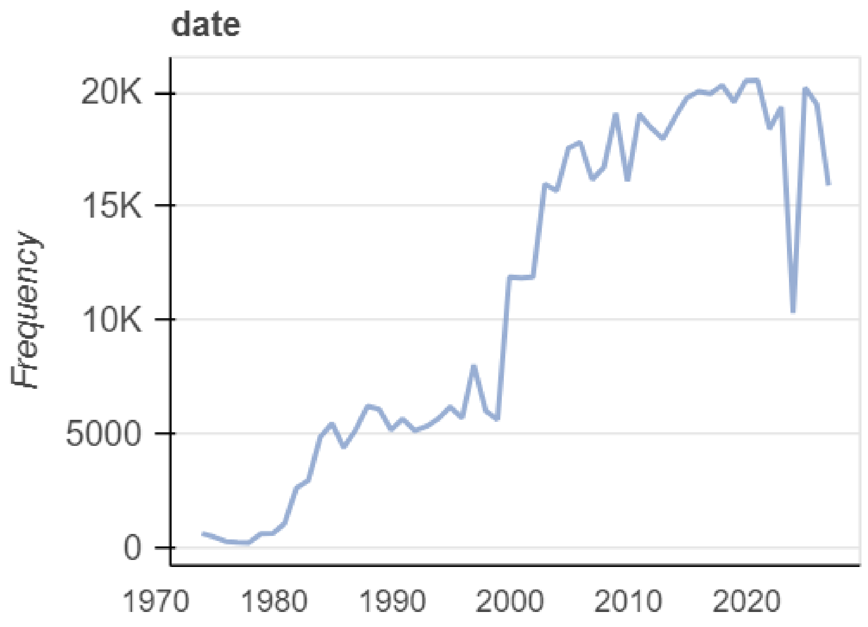
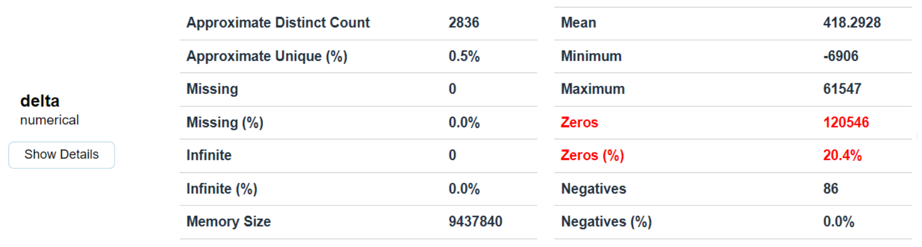


Figure 1: Time trend of races.



delta
numerical
Show Details

Approximate Distinct Count	2836	Mean	418.2928
Approximate Unique (%)	0.5%	Minimum	-6906
Missing	0	Maximum	61547
Missing (%)	0.0%	Zeros	120546
Infinite	0	Zeros (%)	20.4%
Infinite (%)	0.0%	Negatives	86
Memory Size	9437840	Negatives (%)	0.0%

Figure 2: Distribution of `delta` values.

years (see Figure 7). This suggests that filtering the data for more recent years may improve performance, particularly for classification tasks. For instance, the correlation between `points` and `startlist_quality` is more pronounced in recent data see Figure 6. Semantically, the latter makes more sense.

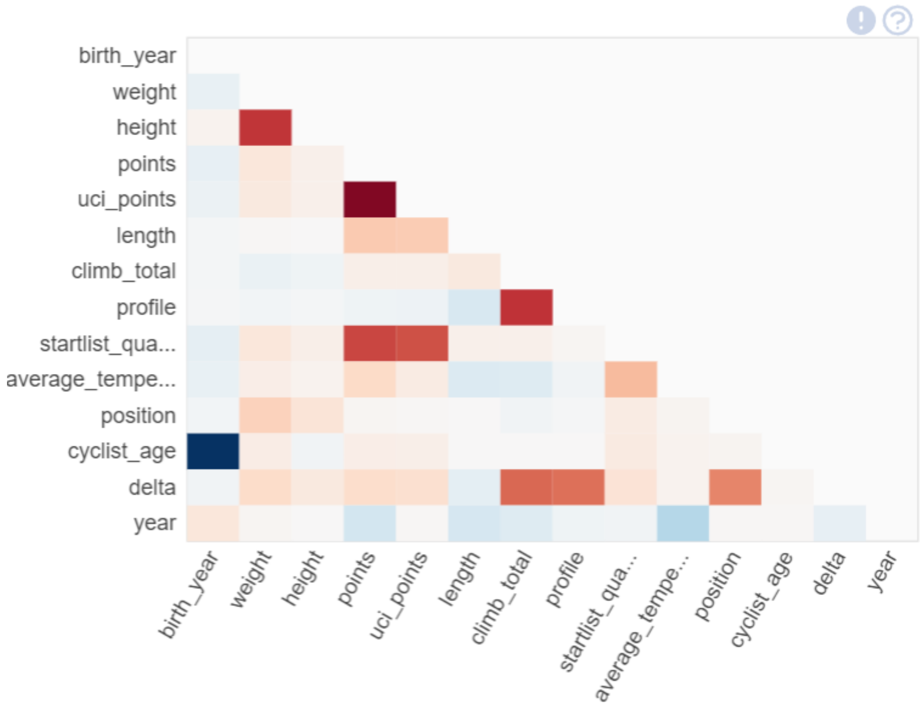


Figure 3: .Spearman correlation coefficients after dropping missing values.

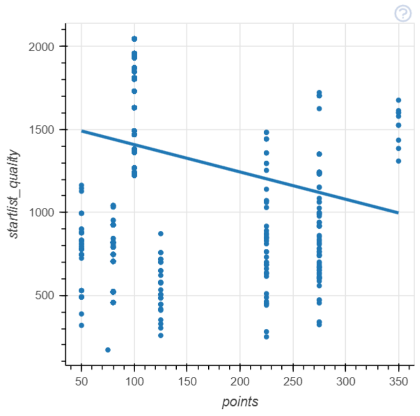


Figure 4: Correlation before 1995.

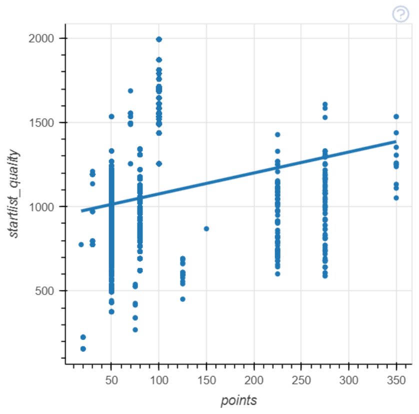


Figure 5: Correlation after 1995.

Figure 6: Comparison of correlations between `startlist_quality` and `points` before and after 1995.

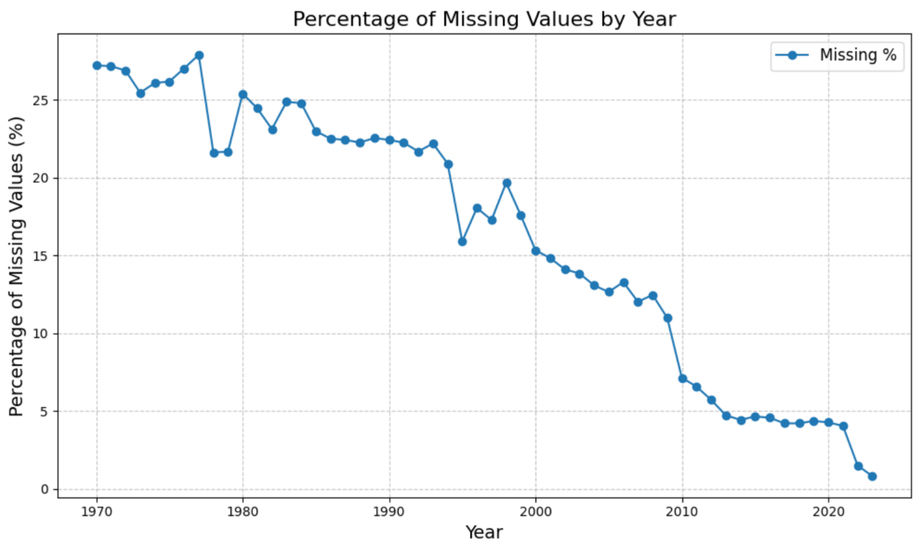


Figure 7: Missing data rate over the years.

2 Data Transformation and Feature Engineering

To clean and prepare the data, several steps are taken to ensure its usability. After that, we aggregate features across races and cyclists in different ways suitable for clustering and classification. This is because in classification we have to be sure that we do not leak the data into present from the future.

Data Cleaning

Rows with negative `delta` values and races with a small number of participants are removed. Additionally, columns such as `average_temperature`, `is_cobbled`, and `is_gravel`, which provide little utility, are dropped. Columns with missing values below a 25% threshold, including `nationality`, `points`, and `age`, are retained but their corresponding rows are dropped. The missing rate calculation for this is not considered by the cyclists dataframe but on the races they took part in because that's the main focus of our analysis for both clustering and classification. As we saw, the older records have higher missing values and are potentially noisier. `birth_year` is dropped as it can be inferred from `age` and `year`.

Data Filtering

The dataset is filtered to include records after 1990, addressing temporal drifts. Prologue stages are excluded due to their distinct dynamics and length, as shown in Figures 8 and 9. They are much shorter, and their deltas are much smaller. Inconsistent `delta` values with respect to positions are interpolated to align with positional data.

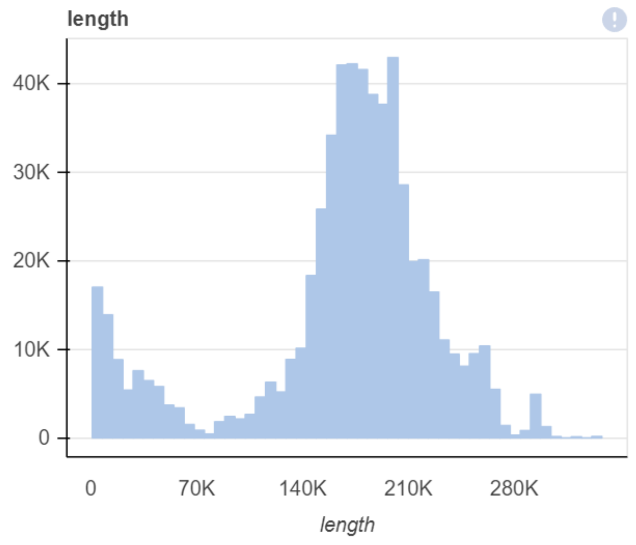


Figure 8: Distribution of race lengths.

Feature Engineering

New basic features are introduced. For example, the number of cyclists in each race and total stages. Positions are scaled by this number to create a score independent of total racers. Similarly, a feature is generated for stages within events to indicate the ordering of stages.

Although the missing rate among cyclists for `weight` and `height` is around half in cyclists, it is not that high when we count missing values in the context of races, since most of the races are recent and cyclists with missing info have fewer races. `Weight` and `height` are imputed based on the mean of their `nationality`, focusing on nations with more than 20 instances (chosen through trial and error by analyzing the distribution of countries). Smaller groups are categorized as "Other" to reduce sampling bias. From the `date` column, only the `month` and `year` are extracted, while other components are discarded. Additionally, the sine and cosine of the month are used for models that handle numerical data to emphasize the circular nature of this feature and make it usable for the models

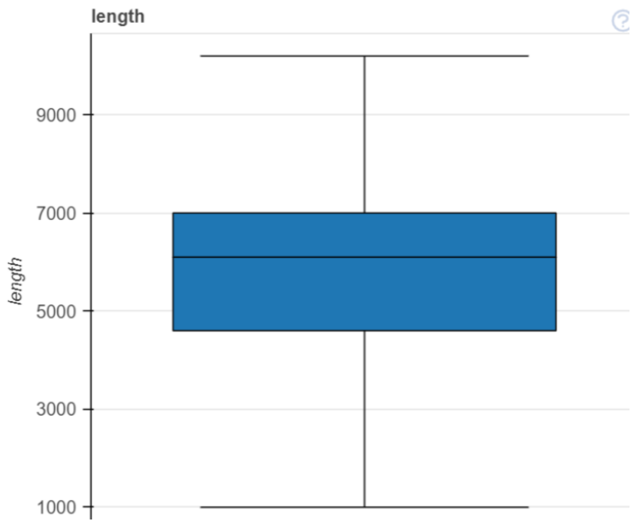


Figure 9: Box plot prologue length.

that can process numerical features only.

$$\text{month\_sine} = \sin\left(\frac{2\pi \cdot \text{month}}{12}\right) \tag{1}$$

$$\text{month\_cosine} = \cos\left(\frac{2\pi \cdot \text{month}}{12}\right) \tag{2}$$

Advanced Features

Highly correlated features like `weight` and `height` are combined into a Body Surface Area (BSA) metric, which is more reflective of overall body size and more suitable for athletes compared to BMI. Similarly, correlations between `profile` and `climb_total` are addressed by imputing missing values through median mapping and finding the nearest point. This process is illustrated in Figures 10 and 11. Remaining instances with missing values in these columns are removed. Missing values in `cyclist_team` are labeled as "Other," including teams with low record counts.

$$\text{BSA} = 0.007184 \cdot \text{weight}^{0.425} \cdot \text{height}^{0.725} \tag{3}$$

There is an interactive EDA dashboard in the main repository named `dashboard1.html` for further exploration.

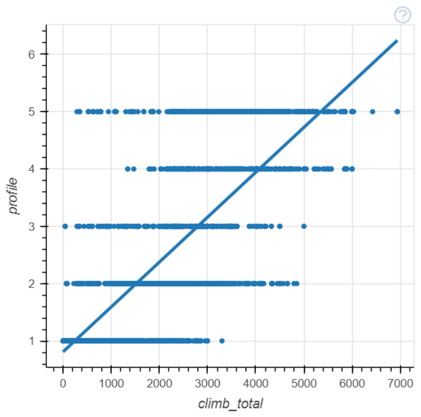
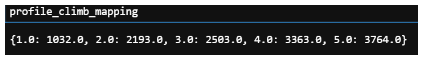


Figure 10: Correlation between profile and climb total.



```
profile_climb_mapping

{1.0: 1032.0, 2.0: 2193.0, 3.0: 2503.0, 4.0: 3363.0, 5.0: 3764.0}
```

Figure 11: Median mapping between profile and climb total.

Outlier Detection

Preliminary outlier detection before aggregation is conducted using Isolation Forest. This step ensures that outliers do not distort feature aggregation. Figure 12 shows the sorted scores. Although the curve looks a bit strange, values higher than 0.48-0.50 seem to be outliers. Initially, we tried to use z-score analysis on the Isolation Forest scores, but they do not appear to follow a normal distribution (see Figure 13). Thus, we decided to go with thresholding. Most of the outliers lie away from the center in the UMAP plot (see Figure 14).

Race Aggregation For Clustering

A diversity index is added for each race, summarizing the variation in `nationality`. The diversity index is computed using Shannon entropy:

$$H = -\sum_i p_i \log(p_i)$$

where p_i is the proportion of cyclists belonging to the i-th nationality within a group defined by `_url`. This metric quantifies the diversity of nationalities in each stage. We attempted to use columns like `age` and `cyclist_team` as well, but the results ended up correlated. The value of this feature increases in recent years, as expected. For numerical columns related to cyclists, including `delta`, `age`, and `BSA`, we compute mean, variance, and skewness as new features for each race.

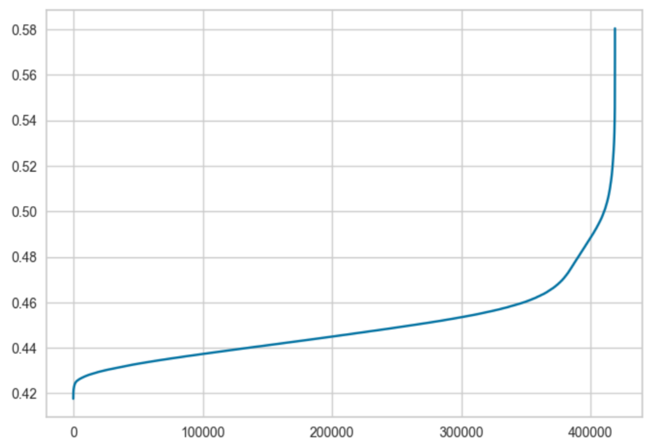


Figure 12: Sorted Isolation Forest scores for outlier detection.

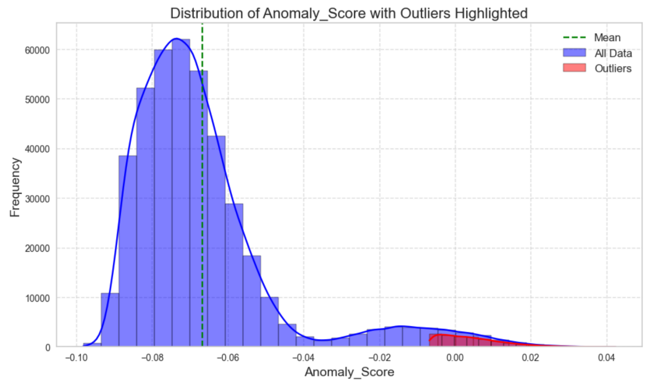


Figure 13: Distribution of Isolation Forest scores.

Aggregation and Encoding for classification

In this part, features from past are aggregated using shifted expanding and rolling means to create meaningful features for races and cyclists performances. First, the diversity index is also used here, too. Given the high cardinality of `nationality`, one-hot encoding is not feasible. Rolling position features including mean and skewness is computed for a window of 3 years for each nation, this is some form of target encoding. Further analysis could be done here on different countries.

”Figure 15” shows that a lot of cyclists participate in many races and are often involved in different race categories. To capture this, we add a feature indicating the number of races each cyclist has participated in so far.

The mean and variance computed earlier for `cyclist_age`, `delta`, and `BSA`

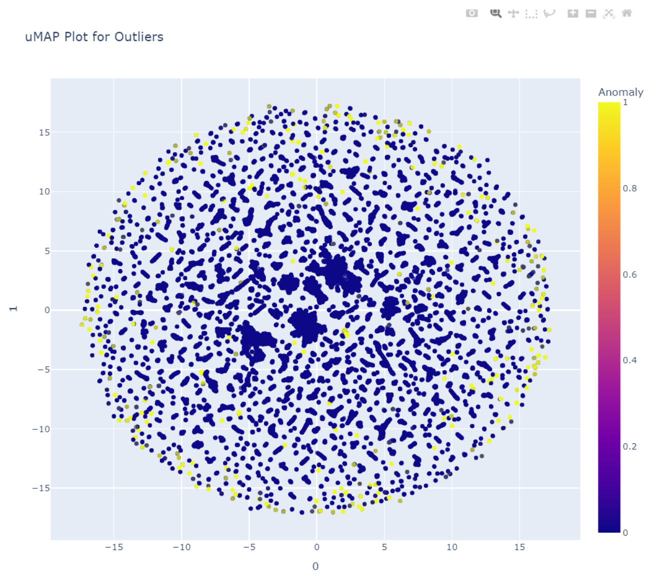


Figure 14: UMAP plot showing the outliers.

are also added. However, `delta` needs to be shifted since we cannot use present data. For this, we compute expanding means shifted by one for many cyclist and race-related features to characterize the types of races and performances the cyclist has at each point in time.

For each race, we also count how many races in the same category the cyclist has participated in to add a sense of experience. Additionally, for important columns such as `position`, lagged values (2, 3) are included as features. To enrich the data further, Exponential Weighted Moving Averages (EWMA) are calculated for many columns within the same `race_category` we are trying to predict. Finally, highly correlated generated features are dropped to avoid redundancy.

3 Clustering

We perform clustering based on races. A collection of features describing each race is selected (most cyclist-related features are dropped). For initial model and feature selection, an extensive search is conducted. In this process, a random subset of the selected features is used. Z-score-based outlier detection is applied (without it, clusters with 1-2 instances and very distant from larger clusters were observed). Additionally, DBSCAN outlier detection is another option. Various models are tested with different hyperparameters (see Figure 16).

Models are ranked based on the SIL score, but the goal is not to choose the one with the best SIL score. Instead, we manually evaluate decent models and feature sets to ensure the features make sense. You can see different features and

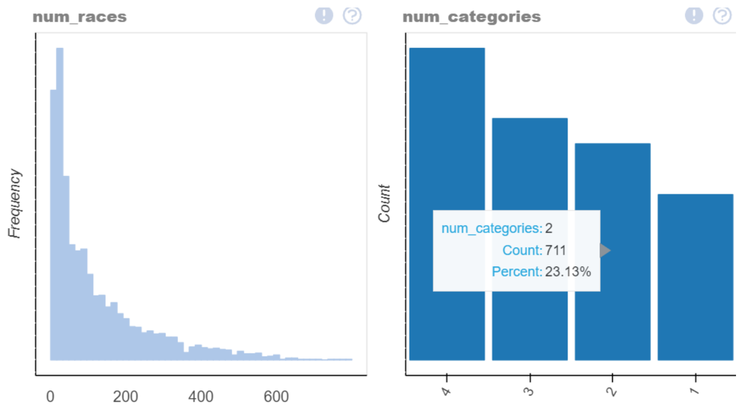


Figure 15: Distribution of races among cyclists across different race categories.

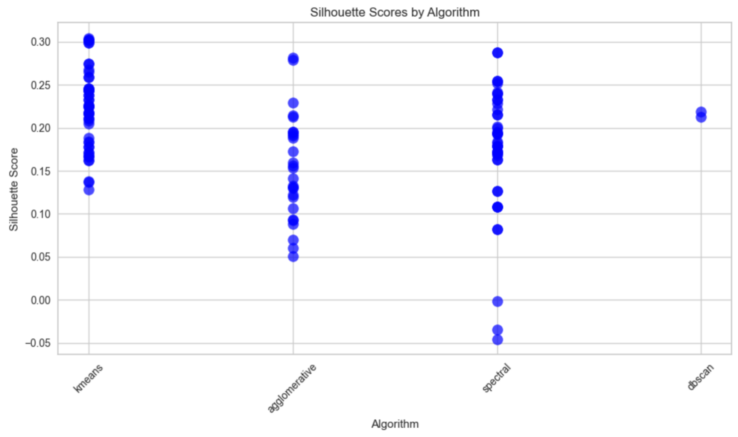
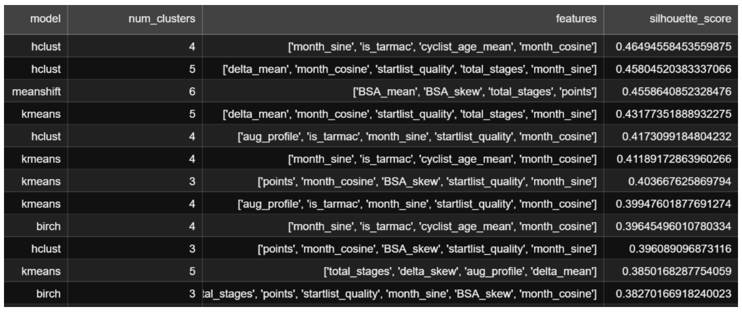


Figure 16: Performance of different clustering models and configurations.

their scores with various models (example in Figure 17). Over 4000 different model and feature configurations were tested. Most performed very poorly. DBSCAN did not yield satisfactory results for the desired number of clusters (3-7). We aimed to keep the number of clusters below 5 or 6 to make the clustering results suitable for feature engineering in the classification problem.

This initial exploration helped identify better models: KMeans and agglomerative clustering. Following this, we used Optuna to tune the selected models and further improve features. Optuna employs advanced sampling algorithms, such as the Tree-structured Parzen Estimator (TPE), for efficient hyperparameter search. An example run for agglomerative clustering can be seen in Figure 18.

After a lot of trial and error and looking at the selected features, some



model	num_clusters	features	silhouette_score
hclust	4	['month_sine', 'is_tarmac', 'cyclist_age_mean', 'month_cosine']	0.46494558453559875
hclust	5	['delta_mean', 'month_cosine', 'startlist_quality', 'total_stages', 'month_sine']	0.45804520383337066
meanshift	6	['BSA_mean', 'BSA_skew', 'total_stages', 'points']	0.4558640852328476
kmeans	5	['delta_mean', 'month_cosine', 'startlist_quality', 'total_stages', 'month_sine']	0.43177351888932275
hclust	4	['aug_profile', 'is_tarmac', 'month_sine', 'startlist_quality', 'month_cosine']	0.4173099184804232
kmeans	4	['month_sine', 'is_tarmac', 'cyclist_age_mean', 'month_cosine']	0.41189172863960266
kmeans	3	['points', 'month_cosine', 'BSA_skew', 'startlist_quality', 'month_sine']	0.403667625869794
kmeans	4	['aug_profile', 'is_tarmac', 'month_sine', 'startlist_quality', 'month_cosine']	0.39947601877691274
birch	4	['month_sine', 'is_tarmac', 'cyclist_age_mean', 'month_cosine']	0.39645496010780334
hclust	3	['points', 'month_cosine', 'BSA_skew', 'startlist_quality', 'month_sine']	0.396089096873116
kmeans	5	['total_stages', 'delta_skew', 'aug_profile', 'delta_mean']	0.3850168287754059
birch	3	tal_stages', 'points', 'startlist_quality', 'month_sine', 'BSA_skew', 'month_cosine']	0.38270166918240023

Figure 17: SIL scores of different features and models.

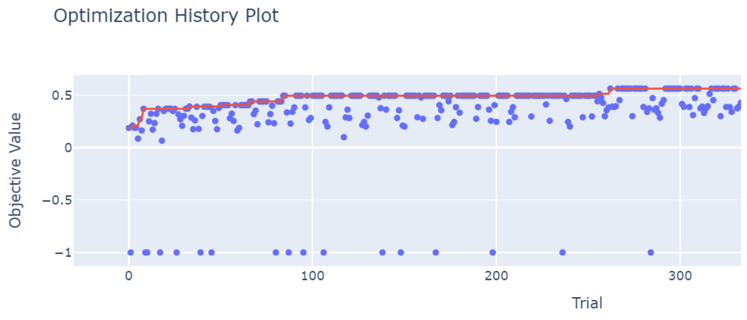


Figure 18: An example Optuna optimization run for agglomerative clustering.

such as `startlist_quality` and `total_stages` appeared over and over in good performing rows. Using Optuna, we specifically tuned models like KMeans and hierarchical clustering while selecting features. For feature selection, each feature is given a binary mask to determine whether it is included. During main model exploration, random subsets of features with a minimum count of four were chosen.

Hierarchical Clustering

We came upon an interesting clustering that used the features `startlist_quality`, `total_stages`, `delta_var`, and `is_tarmac`, with "ward" as the linkage method. Here, `delta_var` represents the variance of delta values for a race. The elbow plot for this clustering is shown in Figure 19. Based on the elbow plot, 5-6 clusters are ideal. However, considering the SIL score (around 1.5 % difference), increasing the number of clusters does not provide significant benefits. Additionally, using fewer clusters simplifies the classification problem, where clustering results encode previous performance. The process is explained in the Feature Engineering section. With more clusters, more instances would need to be dropped. Thus, we opted for 4 clusters.

The mean values for each cluster are shown in Figure 20. One key observation is that clusters differ in `total_stages` (1, 7, or 21). In terms of `startlist_quality`, Cluster 1 stands out as the most prestigious group.

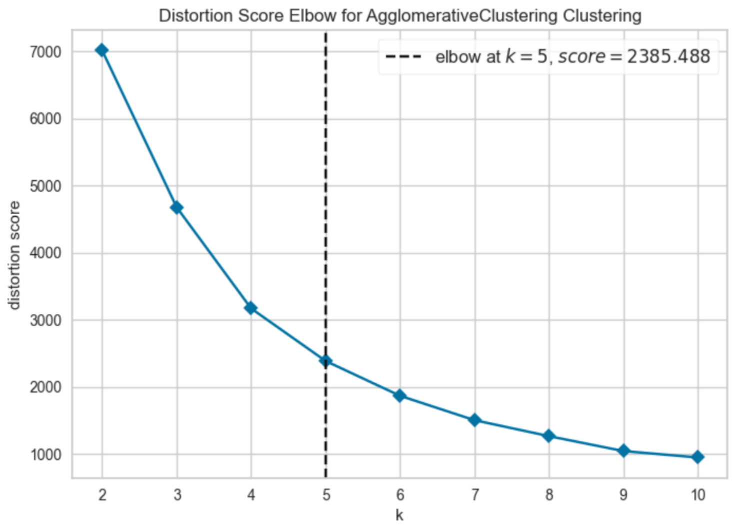
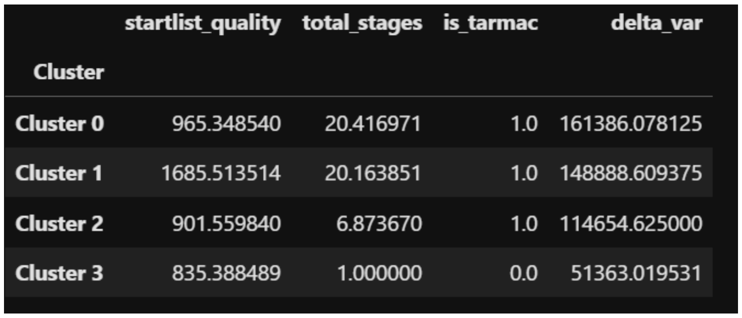


Figure 19: Elbow plot for hierarchical clustering.



Cluster	startlist_quality	total_stages	is_tarmac	delta_var
Cluster 0	965.348540	20.416971	1.0	161386.078125
Cluster 1	1685.513514	20.163851	1.0	148888.609375
Cluster 2	901.559840	6.873670	1.0	114654.625000
Cluster 3	835.388489	1.000000	0.0	51363.019531

Figure 20: Mean values for each cluster.

Further inspection reveals a strong relationship between clustering and the `race_name` feature, which indicates the event name. Clusters often consist of specific events, and events are usually contained within one or two clusters, resulting in a sparse cross-matrix. This is important because the number of race events is much larger than the number of clusters. This observation suggests that different categories of racing events influence race clustering. The mapping of races to clusters is as follows:

- **Cluster 0**: *giro-d-italia* (496), *tour-de-france* (22), *vuelta-a-espana* (578)

- **Cluster 1**: *giro-d-italia* (1), *tour-de-france* (580), *vuelta-a-espana* (11)
- **Cluster 2**: *dauphine* (97), *gran-camino* (7), *itzulia-basque-country* (72), *paris-nice* (132), *tirreno-adriatico* (131), *tour-de-romandie* (62), *tour-de-suisse* (92), *uae-tour* (31), *volta-a-catalunya* (128)
- **Cluster 3**: *amstel-gold-race* (10), *dwars-door-vlaanderen* (14), *e3-harelbeke* (11), *gp-montreal* (10), *gp-quebec* (11), *il-lombardia* (1), *la-fleche-wallone* (21), *liege-bastogne-liege* (4), *omloop-het-nieuwsblad* (16), *paris-roubaix* (2), *ronde-van-vlaanderen* (2), *san-sebastian* (24), *strade-bianche* (10), *world-championship* (3)

Clusters 0 and 1 represent grand tours, the oldest and the most famous events. Cluster 1 is dominated by Tour de France, which has almost twice the `startlist_quality` of other clusters, reflecting its prestige. Cluster 0 mainly consists of the Giro d'Italia and Vuelta a España, both also grand tours. The t-SNE plot (Figure 21) and dendrogram (Figure 22) show that Clusters 0 and 1 are closer compared to the others.

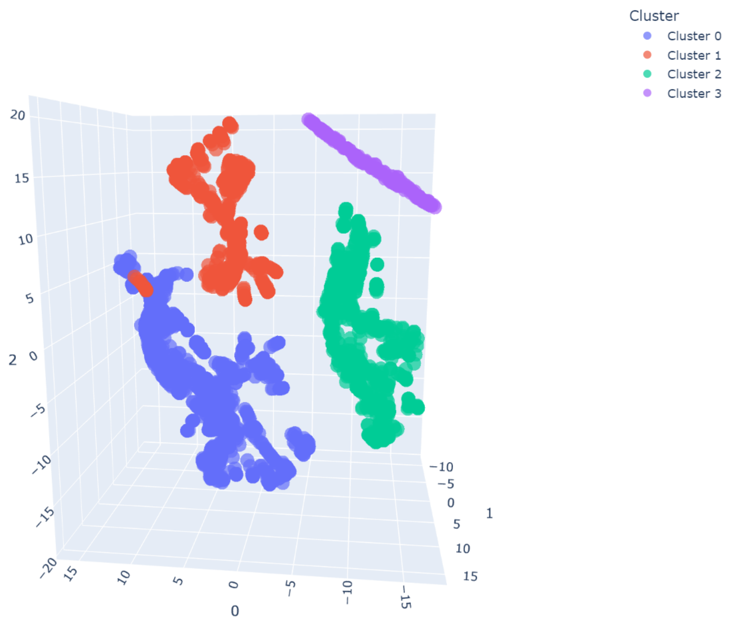


Figure 21: t-SNE plot showing the distribution of clusters.

Cluster 2 represents week-long events often used as preparation for grand tours, while Cluster 3 consists mainly of classic one-day races as part of the UCI World Tour. Clusters 2 and 3 are more recent, as indicated in Figure 23, which shows their distribution over the years. Grand tours generally have larger `total_racers` compared to other clusters, as shown in Figure 24.

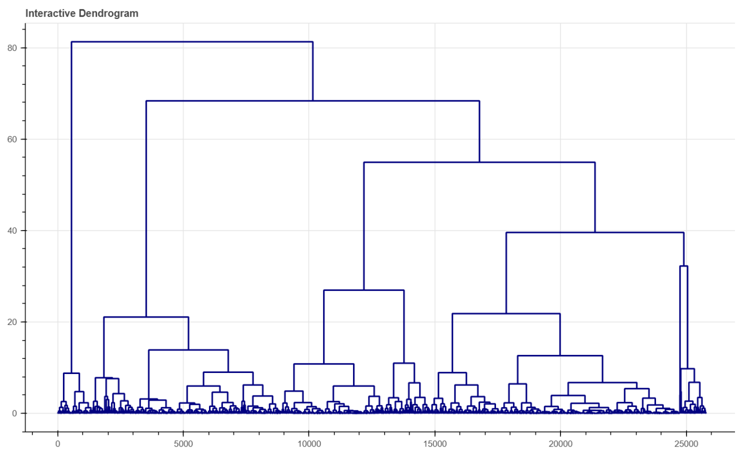


Figure 22: Dendrogram of hierarchical clustering.

KMeans

The best model achieved a SIL score of 0.68 using KMeans with the features `is_tarmac`, `total_stages`, `startlist_quality`, `month_sine`, and `month_cosine`. The cluster feature means are shown in Figure 25, while the normalized centroids are displayed in Figure 26. Compared to hierarchical clustering, this approach provides more seasonal information, indicating which months these events occur. Further time-series analysis is required to explore this seasonality in depth.

4 Classification

If I were to go back, I would definitely use RNNs for this task and treat the data as a time-series for each cyclist. The modeling would be more challenging; however, the feature engineering part would likely be much easier, and the performance could potentially improve. In our current case, we manually aggregate the data through time and do not always identify the best features.

We split the data into training, validation, and test sets. We define the target as being in the top 16% since different races have varying total_racers. This threshold is chosen based on the mean, reflecting the top 20 racers in most races. Normalization and one-hot encoding are applied to the `race_category` feature. For model selection, we use AutoML, testing many models with various hyperparameters. The summary of this process is shown in Figure 27. XGBoost outperformed other models and was selected. Its hyperparameters were tuned using Optuna, optimizing for MCC (since it considers all four elements of the confusion matrix). The best hyperparameter combination is shown in Figure 28. Below are the validation results for the final model.

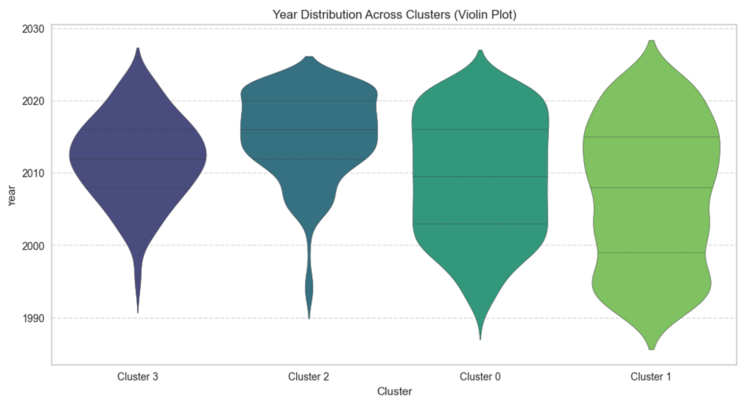


Figure 23: Year distribution for Clusters 2 and 3.

5 Explanation

To handle class imbalance, we tried SMOTE. While it effectively addresses the issue, its usage depends on whether better precision or recall is required, as shown in Figure 32. We ultimately selected XGBoost as the final model. Additionally, changing the classification threshold can help improve performance without using SMOTE.

Initially, we had significant data leakage. By analyzing feature importance plots for linear models and SHAP values, we identified and eliminated the problematic features. The SHAP plot in Figure 34 and feature importance in Figure 33 shows that performance metrics for each cyclist within each category are highly influential features. Obviously, there is more work to be done for the explanation part.

Here are the final results on the test set:

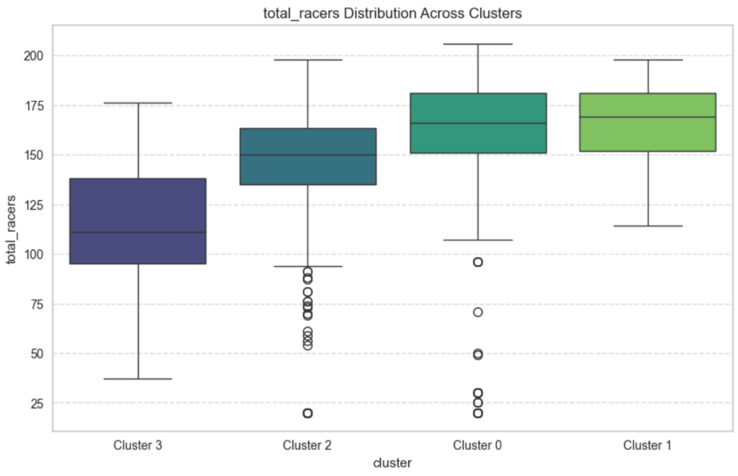
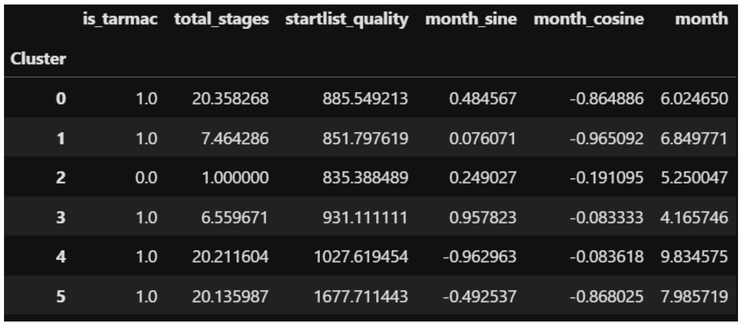


Figure 24: Cluster comparison based on `total_racers`.



Cluster	is_tarmac	total_stages	startlist_quality	month_sine	month_cosine	month
0	1.0	20.358268	885.549213	0.484567	-0.864886	6.024650
1	1.0	7.464286	851.797619	0.076071	-0.965092	6.849771
2	0.0	1.000000	835.388489	0.249027	-0.191095	5.250047
3	1.0	6.559671	931.111111	0.957823	-0.083333	4.165746
4	1.0	20.211604	1027.619454	-0.962963	-0.083618	9.834575
5	1.0	20.135987	1677.711443	-0.492537	-0.868025	7.985719

Figure 25: Feature means for the best KMeans clustering.

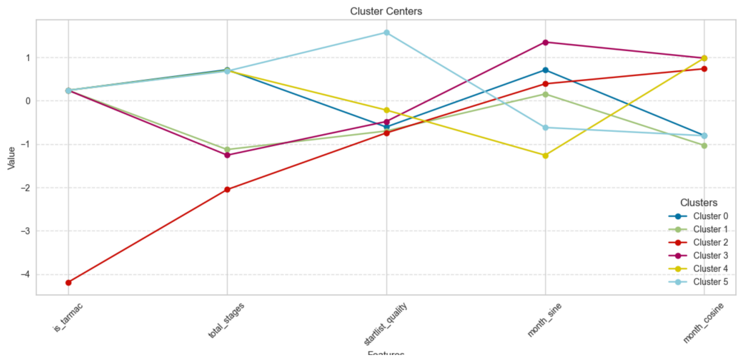
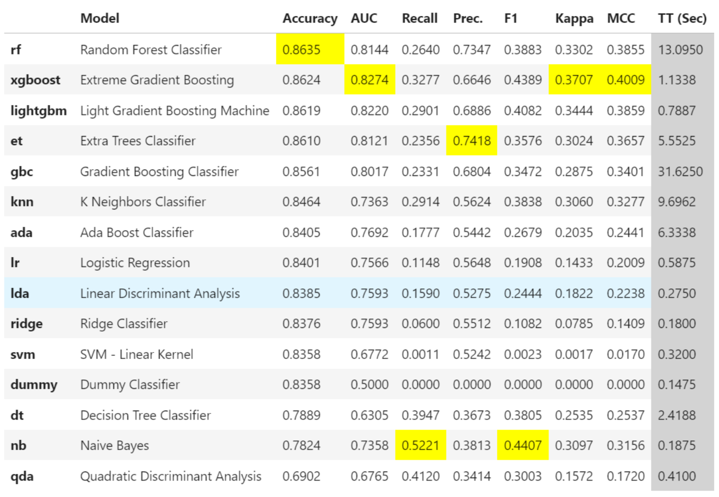
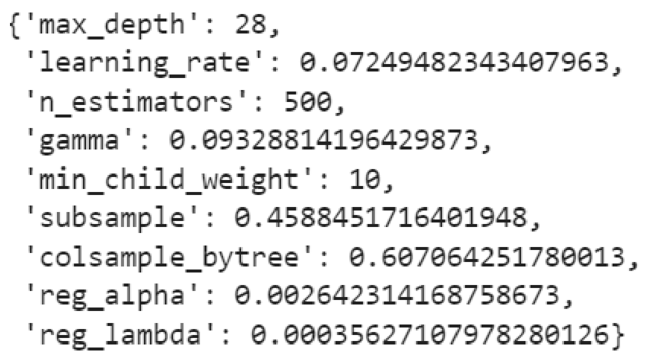


Figure 26: Normalized centroids for the best KMeans clustering.



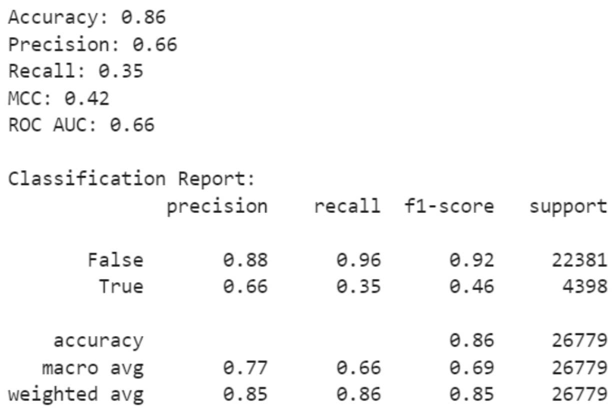
	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.8635	0.8144	0.2640	0.7347	0.3883	0.3302	0.3855	13.0950
xgboost	Extreme Gradient Boosting	0.8624	0.8274	0.3277	0.6646	0.4389	0.3707	0.4009	1.1338
lightgbm	Light Gradient Boosting Machine	0.8619	0.8220	0.2901	0.6886	0.4082	0.3444	0.3859	0.7887
et	Extra Trees Classifier	0.8610	0.8121	0.2356	0.7418	0.3576	0.3024	0.3657	5.5525
gbc	Gradient Boosting Classifier	0.8561	0.8017	0.2331	0.6804	0.3472	0.2875	0.3401	31.6250
knn	K Neighbors Classifier	0.8464	0.7363	0.2914	0.5624	0.3838	0.3060	0.3277	9.6962
ada	Ada Boost Classifier	0.8405	0.7692	0.1777	0.5442	0.2679	0.2035	0.2441	6.3338
lr	Logistic Regression	0.8401	0.7566	0.1148	0.5648	0.1908	0.1433	0.2009	0.5875
lda	Linear Discriminant Analysis	0.8385	0.7593	0.1590	0.5275	0.2444	0.1822	0.2238	0.2750
ridge	Ridge Classifier	0.8376	0.7593	0.0600	0.5512	0.1082	0.0785	0.1409	0.1800
svm	SVM - Linear Kernel	0.8358	0.6772	0.0011	0.5242	0.0023	0.0017	0.0170	0.3200
dummy	Dummy Classifier	0.8358	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1475
dt	Decision Tree Classifier	0.7889	0.6305	0.3947	0.3673	0.3805	0.2535	0.2537	2.4188
nb	Naive Bayes	0.7824	0.7358	0.5221	0.3813	0.4407	0.3097	0.3156	0.1875
qda	Quadratic Discriminant Analysis	0.6902	0.6765	0.4120	0.3414	0.3003	0.1572	0.1720	0.4100

Figure 27: Summary of model performance during AutoML selection.



```
{'max_depth': 28,
 'learning_rate': 0.07249482343407963,
 'n_estimators': 500,
 'gamma': 0.09328814196429873,
 'min_child_weight': 10,
 'subsample': 0.4588451716401948,
 'colsample_bytree': 0.607064251780013,
 'reg_alpha': 0.002642314168758673,
 'reg_lambda': 0.00035627107978280126}
```

Figure 28: Best hyperparameter combination for XGBoost tuning using Optuna.



```
Accuracy: 0.86
Precision: 0.66
Recall: 0.35
MCC: 0.42
ROC AUC: 0.66

Classification Report:
              precision    recall  f1-score   support

       False       0.88      0.96      0.92     22381
        True       0.66      0.35      0.46      4398

    accuracy                           0.86     26779
   macro avg       0.77      0.66      0.69     26779
weighted avg       0.85      0.86      0.85     26779
```

Figure 29: Validation metrics for the XGBoost model.

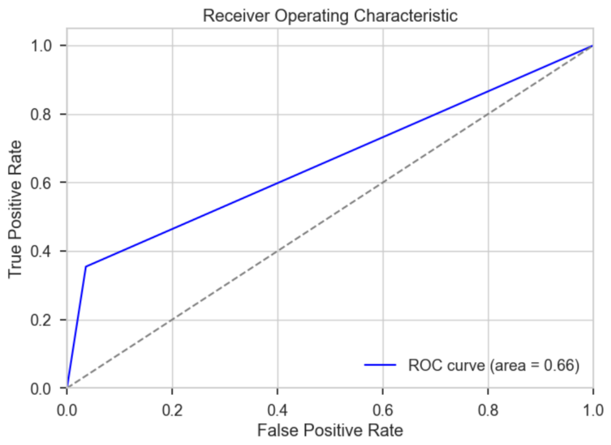


Figure 30: AUC curve for the XGBoost model on validation data.

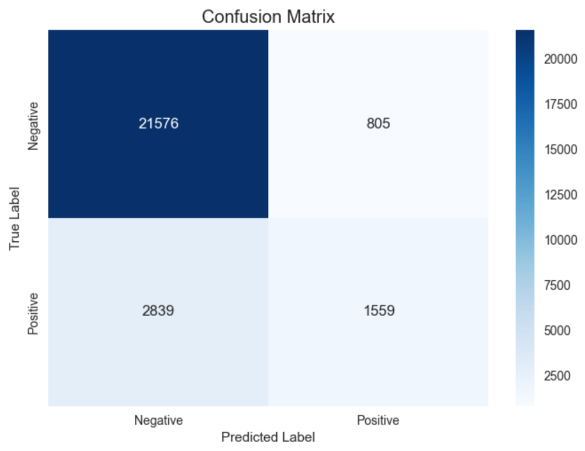
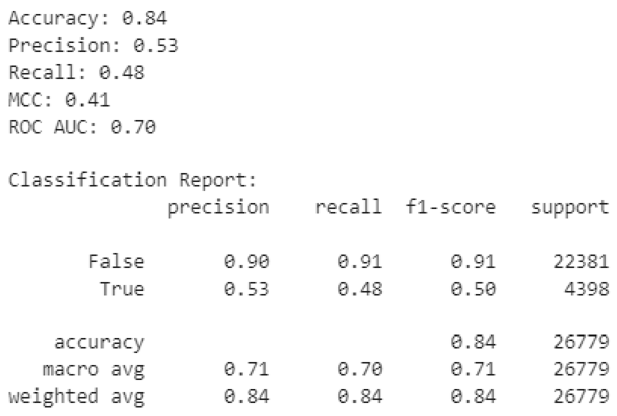


Figure 31: Confusion matrix for the XGBoost model on validation data.



```
Accuracy: 0.84
Precision: 0.53
Recall: 0.48
MCC: 0.41
ROC AUC: 0.70

Classification Report:
              precision    recall  f1-score   support

       False       0.90      0.91      0.91     22381
        True       0.53      0.48      0.50      4398

    accuracy                           0.84     26779
   macro avg       0.71      0.70      0.71     26779
weighted avg       0.84      0.84      0.84     26779
```

Figure 32: Comparison of performance metrics with and without SMOTE.

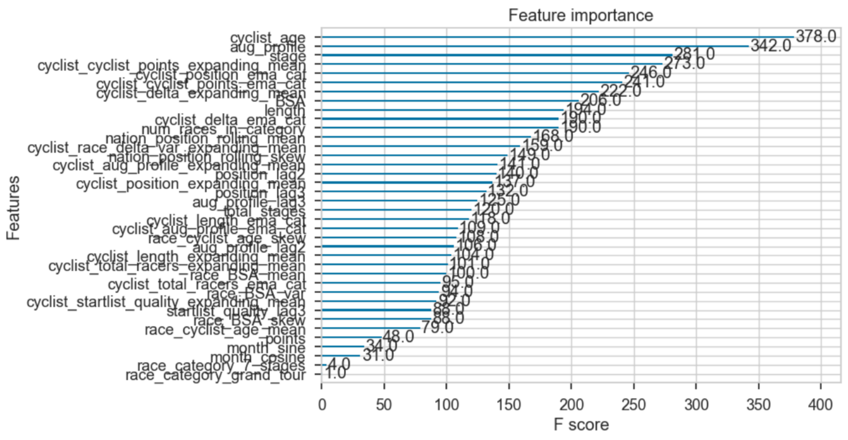


Figure 33: Feature importance plot highlighting the most influential features.

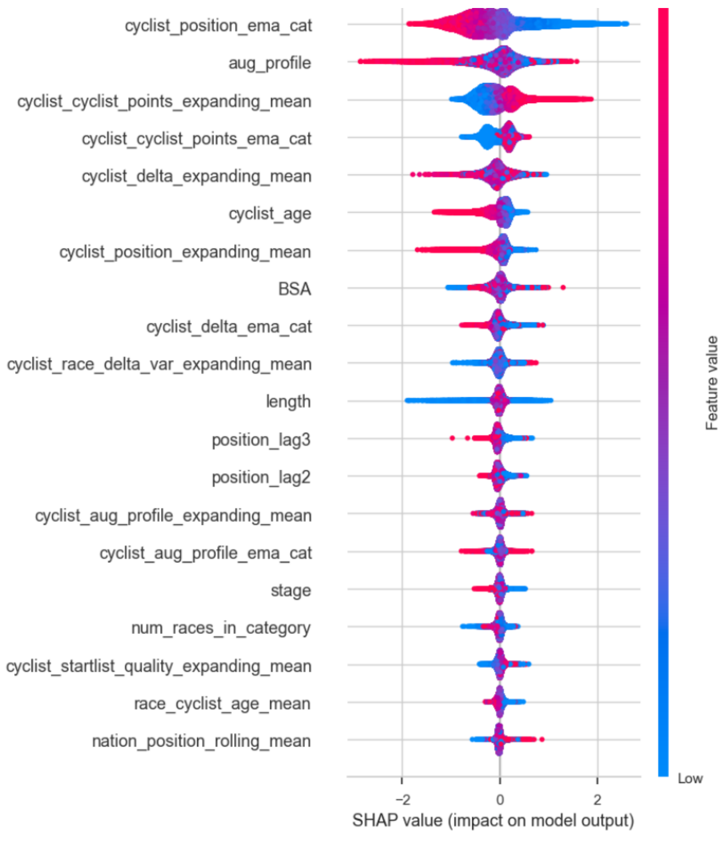
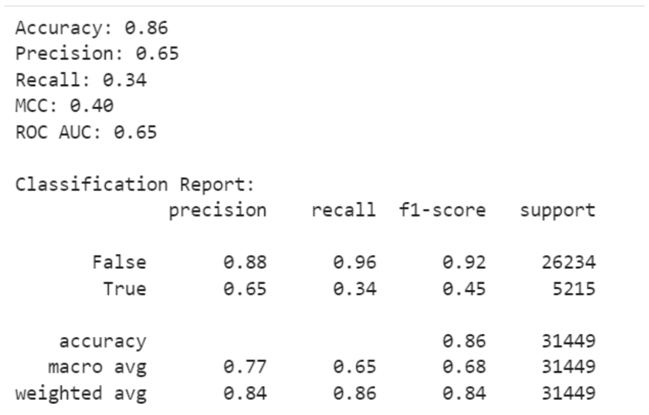


Figure 34: SHAP values highlighting important features.



```
Accuracy: 0.86
Precision: 0.65
Recall: 0.34
MCC: 0.40
ROC AUC: 0.65

Classification Report:
              precision    recall  f1-score   support

       False       0.88      0.96      0.92     26234
        True       0.65      0.34      0.45      5215

    accuracy                           0.86     31449
   macro avg       0.77      0.65      0.68     31449
weighted avg       0.84      0.86      0.84     31449
```

Figure 35: Test metrics for the XGBoost model.