Grundlegende Betrachtung

Der vorliegende Datensatz umfasst 166610 Einträge mit jeweils 12 Attributen. Wir werden im Folgenden ei[1]nige dieser Attribute genauer betrachten.

“TrackId” gibt die Id der Strecke an, auf der das Rennen gefahren wird.
Im Datensatz erscheinen TrackIds von 3 bis 14
Dabei werden jedoch nicht alle Strecken gleich häufig genutzt.
Der Großteil der Rennen findet auf Bahn 12 und 3 statt. Dies bleibt über den gesamten im Datensatz abgebildeten Zeitintervall der Fall.

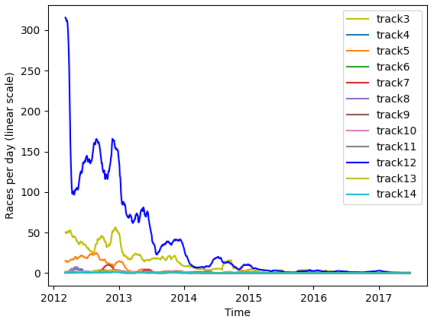

Abb. 1: Zeigt die Anzahl der Rennen pro Tag für jede Rennbahn 20-MA[1]

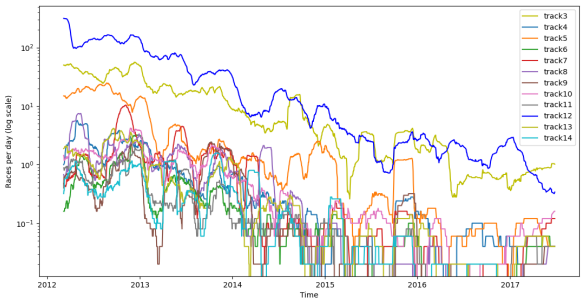

Abb. 2: Zeigt die Anzahl der Rennen pro Tag für jede Rennbahn 20-MA (logarithmisch)

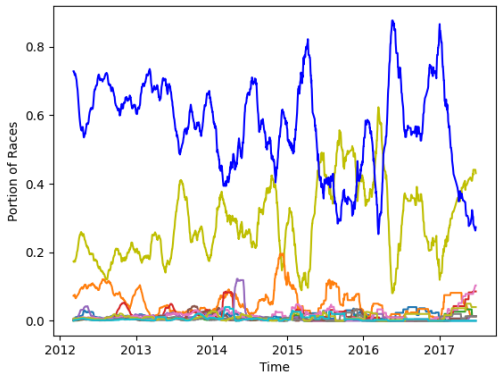

Abb. 3: Zeigt die Anzahl der Rennen pro Tag für jede Rennbahn geteilt durch die Anzahl aller Rennen des Tages 20-MA

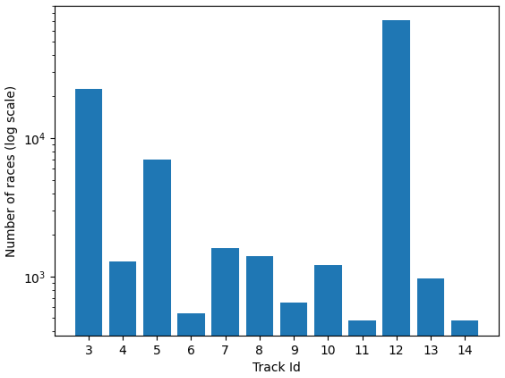

Abb. 4: Anzahl der Rennen für jede Rennbahn (logarithmisch)

“ChallangerId” und “OpponentId” sind die Ids der Fahrer. In dem Datensatz erschienen Ids zwischen 0 und 14664, wobei die 0 einer speziellen Betrachtung bedarf (siehe Fehler).
Der Spieler mit den meisten Rennen hat die Id 48 und fuhr 10461 Rennen.
Der Median der Rennen ist 6 und der Mittelwert ~37,954.

[1] 20-MA: Der Moving Average über 20 Rennen. Dient hier zur Glättung der Kurve.

Die Durchschnittszahl der Tage mit die zwischen erstem und letztem Rennen liegen ist ~84,38 Tage. Der Median ist 6 Tage.
Der Spieler mit der Id 2967 spielte die längste Zeit. Es lagen 1841 Tage zwischen seinem ersten und letzten Rennen. Dabei fand sein erstes Rennen am 09.06.2012 und sein letztes Rennen am 24.06.2017 statt. Über diese Zeit fuhr er jedoch nur 26 Rennen.
Allgemein scheint die Zeit zwischen erstem und letztem Rennen nur geringfügig mit der Zahl der Rennen zusammenhängen. Der Pearson-Koeffizient beträgt 0,245.

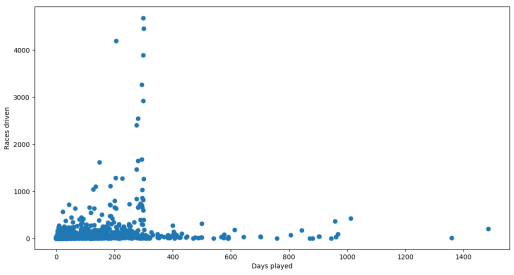

Abb. 5: Zahl der Rennen gegen Zeit zwischen erstem und letztem Rennen

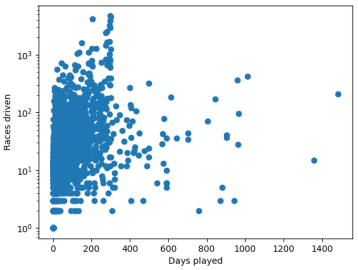

Abb. 6: Zahl der Rennen gegen Zeit zwischen erstem und letztem Rennen (logarithmisch)

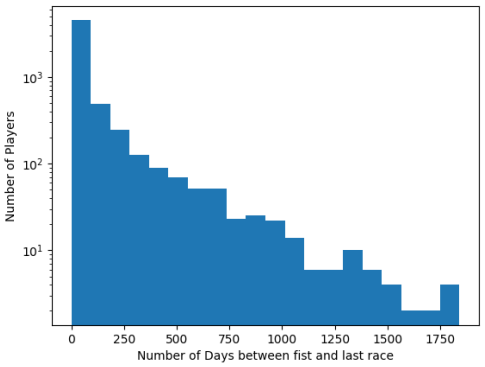

Abb. 7: Histogramm der Zeit zwischen erstem und letztem Rennen (20 bins)

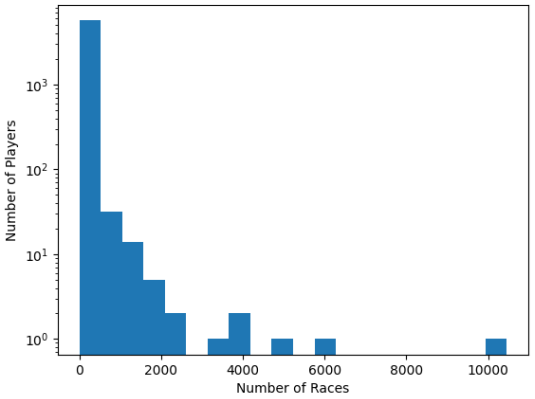

Abb. 8: Histogramm der gesamten Rennanzahl jedes Spielers (20 bins)

“Money” bezeichnet den Einsatz der für die Teilnahme an einem Rennen geleistet werden muss. Der Wertebereich liegt zwischen 30 und 1.000.000. Der minimale Wert im Datensatz ist 30. Es existieren 4 Rennen deren Einsatz über 1.000.000 liegt. Dies wird im Abschnitt Fehlerbehandlung näher behandelt. Der Mittelwert der Einsätze liegt bei 2187,443 und der Median bei 30.
Ein klarer Trend der Einsätze lässt sich in dem betrachteten Zeitintervall nicht beobachten.

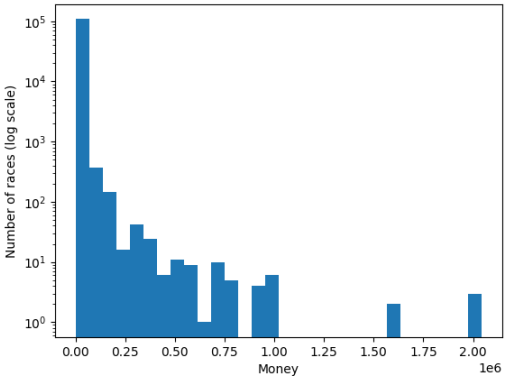

Abb. 9: Histogramm der Renneinsätze (50 bins)

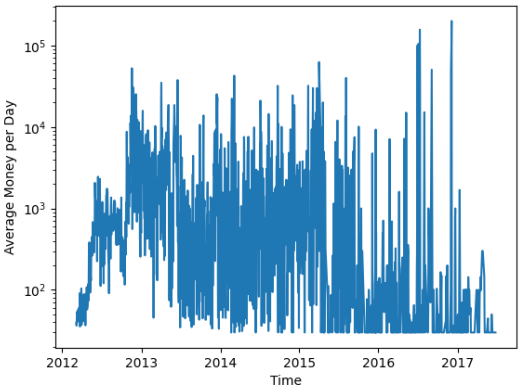

Abb. 10: Durchschnittlicher Renneinsatz pro Tag 20-MA

“Weather” beschreibt das Wetter während des Rennens.
Dabei lautet die prozentuale Verteilung der Wetterausprägungen: rainy: 0,25; snowy: 0,124; sunny: 0,501; thundery: 0,125

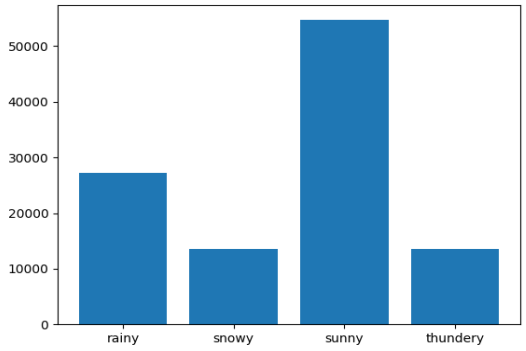

Abb. 11: Häufigkeit der Wetterausprägungen

Weiterhin kann beobachtet werden, dass die Zahl der Rennen mit der Zeit stark abnimmt, was eine verringerte Popularität des Spiels impliziert.
Diese These wird auch durch die verringerte Zahl neuer Spieler gestützt.
Wiederholte Spikes in der Zahl der Rennen und in der Zahl der neuen Spieler könnte auf spezielle Events oder auf Werbeaktionen hinweisen.

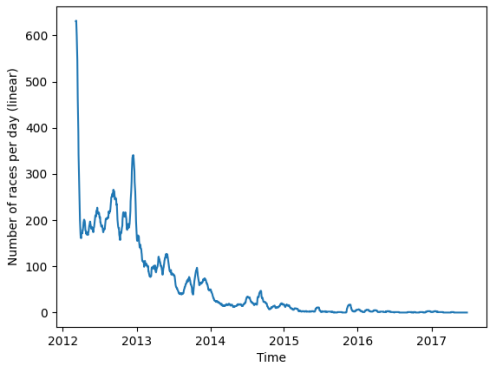

Abb. 12: Rennen pro Tag 20-MA

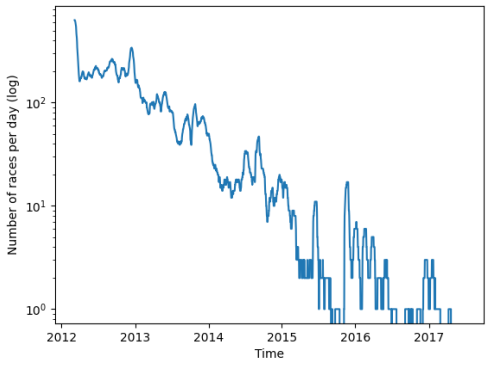

Abb. 13: Rennen pro Tag 20-MA (logarithmisch)

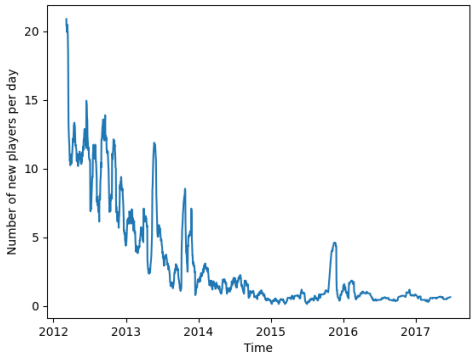

Abb. 14: Anzahl der neuen Spieler pro Tag 20-MA

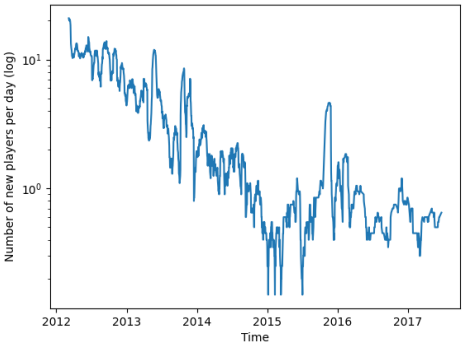

Abb. 15: Anzahl der neuen Spieler pro Tag 20-MA (log)

Fehlerhafte und spezielle Einträge

1. Einheitliche Form der Datumsangaben
   Falls der Status des Rennens nicht ‘finished’ entspricht, so wurde die globale Konstante “0000-00-00 00:00:00” verwendet. Das Format dieser Konstanten unterscheidet sich jedoch von dem Format der Datumsangabe der Rennen mit dem Status ‘finished’ z.B. “06.03.2012 00:00”.
2. Der Wert des Attributes “fuel_consumption” muss 0 betragen, wenn ein Rennen nicht stattgefunden hat. Dies ist bei einigen Rennen jedoch nicht der Fall.
3. Falls der Status eines Rennens nicht ‘finished’ ist, ist der Wert des Attributes Winner 0. Dies impliziert, dass es sich bei der FahrerId 0 um einen globale Konstante handelt, die signalisiert, dass keine valide FahrerId angegeben werden kann. Da für einige Rennen jedoch auch das Attribut Opponent 0 beträgt, ist in diesen Fällen die Id des Herausgeforderten scheinbar unbekannt.
4. Der Wertebereich des Renneinsatzes ist 30 - 1.000.000. Einige Rennen besitzen einen Renneinsatz des außerhalb des Wertebereiches liegen. Die Rennen mit den Ids: 85988; 89636; 89780; 92298; 92302 haben

Renneinsätze die größer als 1.000.000 sind. Hierbei handelt es sich wohl um fehlerhafte Werte.

5. Das Attribut ‘fuel_consumption’ enthält in vielen Fällen Einträge bei denen es sich nicht um eine reelle Zahl handelt. So treten Einträge wie “04. Feb” oder “Apr 35” auf.

Fehlerbehandlung

1. Das abweichende Format der Datumseinträge mit dem Inhalt “0000-00-00 00:00:00” kann leicht dem sonst verwendeten Format angepasst werden. Als globale Konstante wird “01.01.200 00:00” gewählt, da auf diese Weise kein Probleme mit dem Wertebereich des DateTime Datentyps gibt.
2. Die Werte des Attributs “fuel_consumption” kann ohne weitere Probleme auf 0 gesetzt werden.
3. Die Rennen bei denen die OpponentId 0 können für viele Berechnungen unverändert verwendet werden. Lediglich bei Rennen, in welchen OpponentId berücksichtigt wird, sollten diese Rennen aus dem Datensatz entfernt werden. Die OpponentId lässt sich auf keine sinnvolle Weise abschätzen, sodass eine Imputation nicht infrage kommt.
4. Ist ein Renneinsatz größer als der zulässige Wert von 1.000.000 so wird dieser auf 1.000.000 herabgesetzt.
5. Die Werte des Attributs “fuel_consumption” bei denen es sich nicht um eine reelle Zahl handelt werden als Fehlerhaft betrachtet. Dementsprechend werden alle reellen Zahlen als korrekter Wert akzeptiert.
   Bevor mögliche Imputationsverfahren betrachtet werden, wird geklärt, welcher Mechanismus den fehlenden Daten zugrunde liegt.
   Hierzu wird temporär jeder fehlende Wert mit 0 und jeder korrekte Wert mit 1 ersetzt. Anschließend wird überprüft, ob eine Korrelation zwischen den weiteren Attributen eines Rennens vorliegt. Ist dies für mindestens einen Parameter der Fall, so ist der Mechanismus der fehlenden Werte “Missing-At-Random”(MAR) oder “Missing-Not-At-Random”(MNAT).
   Für die Korrelationsanalyse wurde der Chi-Quadrat-Test (falls beide Attribute diskret sind) oder die Pearson Korrelation (falls einer der Parameter kontinuierlich ist) angewandt.
   Auf einem Signifikanzniveau von 0,05 wird eine Korrelation zwischen dem Auftreten eines Fehlers und den Attributen ‘money’, ‘track_id’,’challenger’,’opponent’ und ‘winner’ festgestellt. Somit kann davon ausgegangen werden, dass er Mechanismus “Missing-Not-At-Random” vorliegt.
   Als Nächstes wird untersucht, ob die vorhandenen Werte des Attributes “fuel_consumption” mit dem den weiteren Attributen Korrelieren.
   Hierzu wird die Pearson Korrelation verwendet falls beide Attribute kontinuierlich sind und die Varianzanalyse (ANOVA), falls eines der Attribute diskret ist.

Auf einem 0,05 Signifikanzniveau sind die Werte von “fuel_consumption” abhängig von “track_id”, “challenger”, “opponent”, “winner”, “wheater” und “money”.

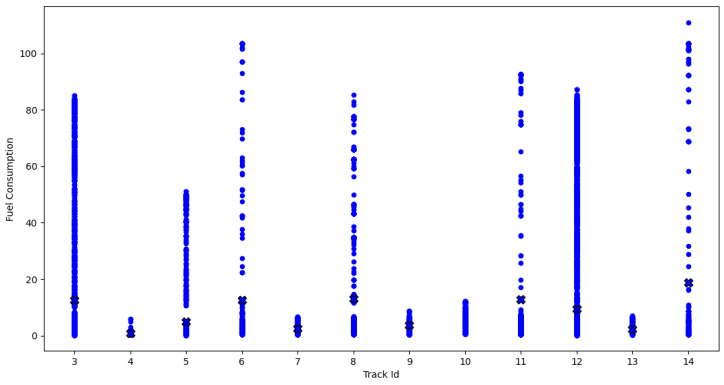

Abb. 16: Treibstoffverbrauch auf Rennbahnen (Kreuze markieren Mittelwerte)

Fazit:
Da der Mechanismus der fehlenden Daten MAR ist, können Rennen mit fehlendem “fuel_consumption”-Wert nicht einfach entfernt werden, da dies einen Bias in den Ausprägungen weiterer Attribute (money’, ‘track_id’,’challenger’,’opponent’ und ‘winner’) hervorrufen würde.
Die mit den Werten von “fuel_consumption” korrelierenden Attribute sollten in die Imputation mit einbezogen werden.
Es wurde der k-Nearest-Neighbour Algorithmus für die Imputation verwendet und dabei die Attribute “track_id”, “challenger”, “opponent”, “winner”, “wheater” und “money” berücksichtigt. Als k-Wert wurde 3 gewählt (dies begründet sich durch die steigende benötigte Rechenleistung bei größeren k-Werten. Eine verbesserte Genauigkeit der Imputation bei größeren k-Werten ist jedoch zu erwarten).

Klassifikation

Nachdem der Datensatz von fehlerhaften Daten befreit wurde, ist eine Feature-Konstruktion für eine Vorhersage des Rennausgangs möglich.

Eine Vorhersage des Rennausgangs anhand der Fahrer-Id ist nur wenig sinnvoll, da für einige Fahrer nur eine sehr kleine Zahl an Rennen vorliegt. Auch ist davon auszugehen dass sich die Leistungsfähigkeit eines Spielers mit der Zeit verändert, während er mehr Erfahrung im Spiel sammelt.
Stattdessen sollte die Leistung während der, in der Vergangenheit liegenden Rennen betrachtet werden.

Die abgesehen von ‘money’ und ‘weather’ können keine Attribute ohne weitere Bearbeitung für die Konstruktion eine Trainingsdatensatzes genutzt werden. Die

weiteren Features werden aus den in der Vergangenheit liegenden Rennen des Herausforderers und des Herausgeforderten ermittelt.

1. Anteil der gewonnen Rennen bezogen auf die letzten n Rennen.
2. Gewonnenen Rennen minus der verlorenen Rennen geteilt durch n.
3. Anteil der Rennen, die auf der gleichen Strecke stattgefunden haben bezogen auf die letzten n Rennen.
4. Durchschnittlicher Treibstoffverbrauch der letzten n Rennen.

Dabei werden für n 10; 25 und 50 gewählt und die Features für den Herausgeforderten und den Herausforderer berechnet.

Falls der Herausforderer das Rennen gewinnt, wird als Label 0 eingefügt andernfalls wird das Label 1 eingefügt. In dem Datensatz existieren 61179 Rennen mit Label 0 und 47920 Rennen mit dem Label 1.

Der XGBoost Algorithmus wurde verwendet, um Entscheidungsbäume für die Klassifikation zu erhalten. Dabei wurden 80% Prozent des Datensatzes zum Training und 20% zur Evaluation verwendet. Nach einer linearen Suche nach den optimalen Werten für die Anzahl der Bäume und ihre maximale Tiefe wurde eine Genauigkeit von 0,755 erzielt. Hierfür wurde 99 Bäume mit einer maximalen Tiefe von 15 verwendet.

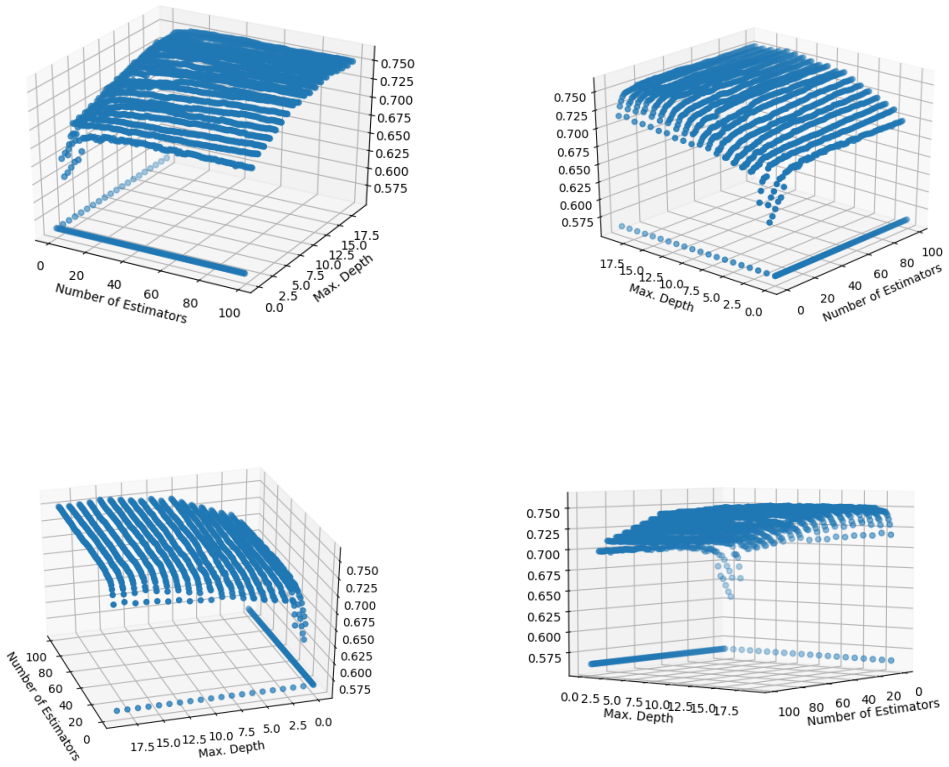

Abb. 17: Ergebnis der Hyperparameter Optimierung mit Genauigkeit auf der z-Achse.

Bedienung:

testMARvsMCAR
Überprüft ob der Mechanismus der Fehlenden Daten MAR oder MCAR ist.

python3 IT-Wettbewerb.py “testMARvsMCAR”

testFuelConsumptionCorrelation
Überprüft Korrelationen zwischen “fuel_consumption” und anderen Spalten.

python3 IT-Wettbewerb.py “testFuelConsumptionCorrelation”

imputeData
Schätzt fehlerhafte Werte in der Spalte “fuel_consumption” mittels k-Nearest-Neigbour Algorithmus ab und setzt sie ein.
(Diese Methode läuft sehr lang und benötigt viel Arbeitsspeicher zum testen sollte daher nur ein Teil der “race.csv” Datei verwendet werden)

python3 IT-Wettbewerb.py “imputeData”

createFeatures
Erzeugt Features die zur Vorhersage von Renngebnissen verwendet werden können.
(Diese Methode läuft sehr lang und benötigt viel Arbeitsspeicher zum testen sollte daher nur ein Teil der “race.csv” Datei verwendet werden)

python3 IT-Wettbewerb.py “createFeatures”

plotRaces
Zeigt die Zahl der Rennen pro Tag im Verlauf der Zeit.
Glättung mittels Moving Average über N (standardmäßig 20) Tage.

python3 IT-Wettbewerb.py “plotRaces”
oder
python3 IT-Wettbewerb.py “plotRaces” N
z.B.
python3 IT-Wettbewerb.py “plotRaces” 20

plotTopTrack
Zeigt den Anteil der Rennen pro Rennbahn im Verlauf der Zeit.
Glättung mittels Moving Average über N (standardmäßig 20) Tage.
Abbildung kann stationär gewählt werden wenn b=1 (standardmäßig 0)

python3 IT-Wettbewerb.py “plotTopTrack”
oder
python3 IT-Wettbewerb.py “plotRaces” N b
z.B.
python3 IT-Wettbewerb.py “plotRaces” 20 1

getAverageFuelConsumption
Gibt den Durchschnitt der Spalte “fuel_consumption” in der Konsole aus.

python3 IT-Wettbewerb.py “getAverageFuelConsumption”

plotPlayDuration
Zeigt ein Histogramm der Zeiten zwischen erstem und letztem Rennen jedes Spielers.

python3 IT-Wettbewerb.py “plotPlayDuration”

plotPlayerNumRaces
Zeigt ein Histogramm der Gesamtzahl der Rennen eines jeden Spielers.

python3 IT-Wettbewerb.py “plotPlayerNumRaces”

plotNewPlayerRate
Zeigt Anzahl der neuen Spieler pro Tag über die Zeit.
Y-Achse kann logarithmisch angezeigt werden falls b=1

python3 IT-Wettbewerb.py “plotNewPlayerRate”
oder
python3 IT-Wettbewerb.py “plotNewPlayerRate” 1

plotWeather
Zeigt die Häufigkeit der Wetterausprägungen.

python3 IT-Wettbewerb.py “plotWeather”

plotMoney
Zeigt Histogramm der Renneinsätze

python3 IT-Wettbewerb.py “plotMoney”

plotFuelConsumptionAgainstTrackID
Zeigt Treibstoffverbrauch für jede Rennbahn

python3 IT-Wettbewerb.py “plotFuelConsumptionAgainstTrackID”

train
Trainiert den XGBoost Algorithmus auf die Vorhersage von Rennenergebnissen.
Mit den Daten in dataDir und speichert das Modell in modelDir.

python3 classify.py “train”
oder
python3 classify.py “train” “modelDir” “dataDir”

predict
Gibt vorhergesagte Rennergebnisse aus.
Mit dem Modell in modelDir und den Daten n dataDir.

python3 classify.py “predict”
oder
python3 classify.py “predict” “modelDir” “dataDir”

Setup

1. Repo herunterladen
2. Entpacke data.zip in Ordner main

Dependencies

numpy version 1.21.4
pandas version 1.3.4
matplotlib version 3.5.0
scipy version 1.7.3
scikit-learn version 1.0.1
xgboost version 1.5.1