Vorhersagen verschiedener Formel 1 Events

Fiete Scheel

28. Januar 2026

Inhaltsverzeichnis

1 Einleitung

1.1 Hintergrund und Problemstellung

Die Formel 1 ist ein stark datengetriebener Sport, weshalb sich die Leistung von Fahrern und Teams gut analysieren lässt. Die zentrale Frage dieser Arbeit ist, ob man aus den vor einem Rennen verfügbaren Informationen zuverlässige Ergebnisse vorhersagen kann. Das Problem wird dabei zweigeteilt: Zunächst wird klassifiziert, ob ein Fahrer überhaupt Punkte erzielt. Darauf aufbauend wird der zeitliche Rückstand zum Sieger modelliert, da dieser Wert eine stabilere Zielvariable darstellt als die durch äußere Einflüsse oft verfälschte absolute Rennzeit.

1.2 Zielsetzung und Ressourcen

Das Ziel dieser Arbeit ist der Aufbau eines reproduzierbaren Datensatzes sowie die Entwicklung und der Vergleich geeigneter Machine-Learning-Modelle. Unter Einbeziehung verschiedener Merkmale, wie etwa der Startposition oder des Streckentyps, werden die Modelle trainiert und anschließend evaluiert. Dies dient als Basis, um perspektivisch auch Podiumsplatzierungen vorhersagen zu können. Der gesamte Code, der in diesem Projekt verwendet wurde, ist unter https://github.com/mcfiet/dl-project zu finden.

2 Stand der Forschung

In der Forschung gibt es bisher wenige Arbeiten, die sich mit der Vorhersage von Rennergebnissen in der Formel 1 beschäftigen. Es gibt einige private Projekte und Blogs, die sich mit diesem Thema auseinandersetzen (vgl. Mehta, 2019), jedoch fehlt es an wissenschaftlichen Veröffentlichungen, die systematisch verschiedene Ansätze vergleichen und evaluieren. Einige Lösungen nutzen jedoch vergleichbare Datensätze und Merkmale, wie sie in dieser Arbeit verwendet werden. Oft wurde das Problem mit einem Random Forest Classifier (vgl. Roger, 2020) oder Gradient Boosting Modellen (vgl. Antaya, Mariana, 2025) angegangen, da diese Modelle gut mit tabellarischen Daten umgehen können und weniger anfällig für Overfitting sind.

3 Methodik

3.1 Datengrundlage

Als Datenquelle dient das Python-Paket FastF1 (Contributors, 2024), das offizielle F1-Ergebnis- und Qualifyingdaten bereitstellt. Pro Saison und Rennen werden Qualifying- und Rennsessions geladen und pro Fahrer ein Datensatz erzeugt; jede Zeile entspricht damit einem Fahrer in einem Grand Prix. Zwei Builder-Skripte erzeugen die Features:

- **Klassifikation (`build_fastf1_dataset.py`):** Verwendet ausschließlich vor dem Rennen verfügbare Informationen zur Vorhersage `points_scored`.

Merkmale: Fahrer-, Team- und Strecken-IDs (`driver_id`, `constructor_id`, `circuit_id`), Startposition (`grid_position`), Qualifying-Deltas (gesamt und zum Teamkollegen: `quali_delta`, `quali_tm_delta`), kumulierte Saisonpunkte (`season_pts_driver`, `season_pts_team`), Punktemittelwert der letzten drei Rennen (`last_3_avg`) sowie Kontextflags für Stadtkurse und Regen (`is_street_circuit`, `is_wet`).

- **Regression (`build_fastf1_dataset_regression.py`):** Rekonstruiert konsistente Rennzeiten, indem die Siegerzeit als Referenz genutzt und Gaps addiert werden; DNFs bleiben als Status erkennbar. Zusätzliche Ziele: `gap_to_winner` (Sieger=0), `race_time_per_lap` und Rundenanzahl (`laps`).

Um Generalisierung auf neue Fahrer/Teams/Strecken zu prüfen, werden unbekannte Kategorien optional maskiert. Jahressplits verhindern, dass spätere Saisons in die Trainingsphase leaken. Die Ausfallquote pro Saison wurde mit dem Skript `calc_retirement_rate.py` aus den FastF1-Starts berechnet.

3.2 Modellarchitekturen

Die Daten sind tabellarisch mit einer Mischung aus kategorischen und numerischen Merkmalen. Für beide Aufgaben (Klassifikation und Regression) werden mehrere Modellfamilien verglichen:

- **Lineare Baselines:** Logistische Regression (Klassifikation) bzw. lineare Modelle mit Regularisierung (Regression) als Vergleich.
- **Lineare Baseline:** Random Forests als robuste, nichtlineare Baseline (Breiman, 2001).
- **Gradient Boosting:** XGBoost (Chen und Guestrin, 2016) und LightGBM (Ke et al., 2017) für tabellarische Daten; effizient in Interaktionen und nichtlinearen Effekten.
- **Neuronale Netze:** Mehrschichtige Perzeptren mit Early Stopping, hauptsächlich für die Klassifikation erprobt.
- **TabPFN:** Vortrainiertes Few-Shot-Transformer-Modell (Hollmann et al., 2022), das ohne Hyperparameter-Tuning konkurrenzfähige Performance liefern kann. TabPFN ist speziell für kleine bis mittlere tabellarische Datensätze ausgelegt; mit 3740 Trainingsbeispielen (insgesamt 4698 Zeilen) und 12 Merkmalen (4 kategorial, 8 numerisch) liegt der verwendete Datensatz im Zielbereich dieses Modells.

3.3 Trainingssetup

Die Datensätze werden nach Saison getrennt, um zeitliche Leckage zu vermeiden. Typische Splits: Training bis einschließlich 2023, Validierung 2024, Test 2025 für Regression; für die Klassifikation wurden auch Saisons 2015–2025 in Train/Val/Test mit 3740/479/479 Beispielen genutzt. Kategorische Variablen werden je nach Modell

One-Hot-kodiert oder als Integer-Indizes verwendet. Numerische Merkmale werden imputiert (Median) und skaliert. Für unausgewogene Klassen kommt in Baumverfahren eine balancierte Gewichtung zum Einsatz. Allerdings hat sich gezeigt, dass die Klassenungleichheit bei der Klassifikation fast nicht existent ist, da durch die Punkteregelung fast die Hälfte der Fahrer Punkte erzielt.

Modellselektion erfolgt auf dem Validierungsset. Bei der Klassifikation wird der Entscheidungsschwellenwert auf den Validierungs-*F1*-Score optimiert und anschließend auf dem Testset ausgewertet. Die Metriken die verwendet wurden: *F1* binär/-makro und *Balanced Accuracy*. In der Regression stehen *MAE*/*RMSE* im Fokus; da ich zuerst davon ausgegangen wurde, dass sehr kleine Gaps prozentuale Kennzahlen aufblähen, wurde *MAPE* nur ab 5 s Gap sowie *SMAPE* ergänzend angegeben. Allerdings hatte das keinen großen Einfluss auf *MAPE*.

3.4 Inferenz und UI

Für die Klassifikationsaufgabe wurde ein leichtgewichtiges Inferenz-Setup ergänzt: Das beste Modell (TabPFN) wird nach dem Fit inklusive der Kategoriekodierung gespeichert und über einen kleinen API-Endpunkt bereitgestellt. Darauf aufbauend wurde eine React-Anwendung mit Material UI umgesetzt, in der die Featurewerte eingetragen werden können und die anschließend die vorhergesagte Klasse sowie die zugehörige Wahrscheinlichkeit ausgibt. Die Oberfläche dient als praktische Demonstration, wie das trainierte Modell mit den im Notebook definierten Features (Fahrer, Team, Strecke, Jahr und numerische Kontextmerkmale) in eine bedienbare Anwendung überführt werden kann. Die Anwendung kann unter folgender Adresse getestet werden: https://dl.devoniq.de/. Da dies ein Server ist, der primär für andere Zwecke genutzt wird, kann es für eine Vorhersage bis zu 5min dauern.

4 Experimente

4.1 Klassifikation: Punktevorhersage

Die Notebook-Serie `notebooks/points_scored/` dokumentiert die Experimentreihen zur Klassifikation und folgt den in Abschnitt *Modellarchitekturen* beschriebenen Familien: lineare Baselines, baumbasierte Verfahren (Random Forest, XGBoost), ein MLP sowie TabPFN. Die Umsetzung liegt in `baseline_training.ipynb`, `random_forest_baseline.ipynb`, `xgboost_training.ipynb`, `model_training.ipynb` und `tabpfn.ipynb`; ergänzend analysiert `learning_curve_analysis.ipynb` die Abhängigkeit von der Trainingsgröße.

Im Laufe der Experimente wurden außerdem verschiedene Feature-Engineering-Ansätze mit dem TabPFN getestet. Dazu zählten Versuche mit potenziell "leckenden" Merkmalen wie `avg_racetime`, die später wieder entfernt wurden, sowie Variationen im Encoding (One-Hot, Integer-Indizes, Target Encoding) und das Maskieren unbekannter Kategorien.

Die Analyse zeigt deutlich, dass die Startposition (`grid_position`) der wichtigste Faktor für die Vorhersage ist: Je weiter vorne ein Fahrer startet, desto

wahrscheinlicher erzielt er Punkte. Als zweitwichtigste Indikatoren identifiziert das TabPFN-Modell die aktuelle Form (`last_3_avg`) und die Saisonleistung des Teams (`season_pts_team`).

Auffällig ist, dass die Team-Punkte wichtiger eingestuft werden als die Punkte des Fahrers. Da beide Werte extrem stark zusammenhängen (Korrelation von 0.96), hat das Modell gelernt bevorzugt die Team-Daten zu nehmen. Reine Identifikationsmerkmale wie die `constructor_id` liefern keinen Mehrwert und wirken sich sogar leicht negativ auf die Vorhersagequalität aus.

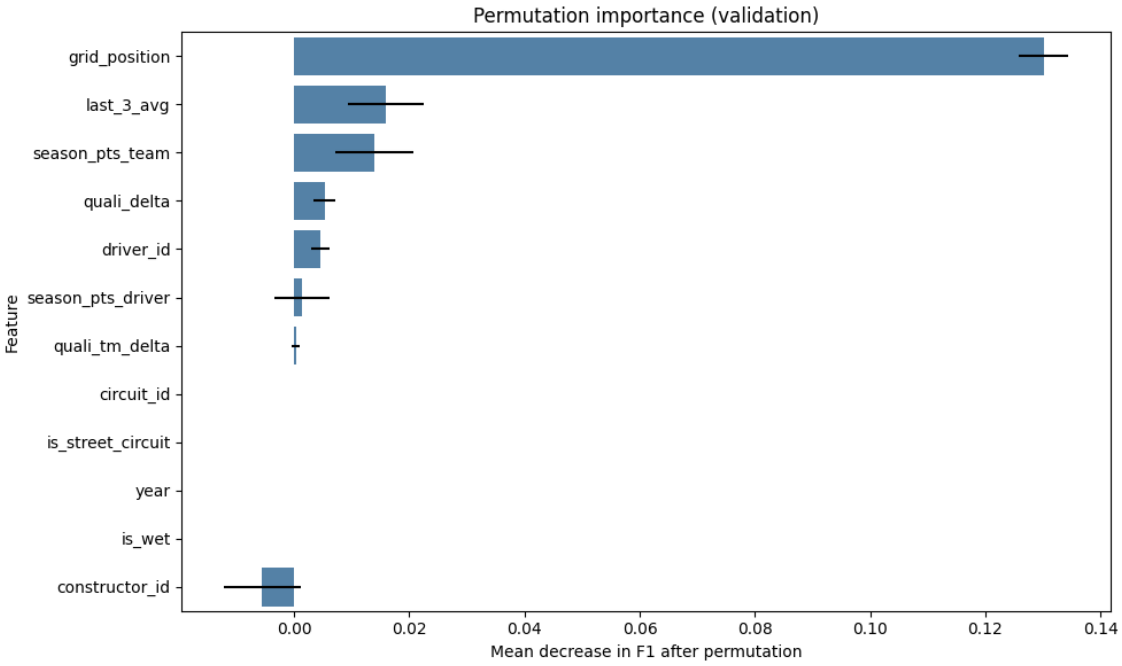

Abbildung 1: Feature Importance des TabPFN-Modells für die Klassifikationsaufgabe (Punktevorhersage).

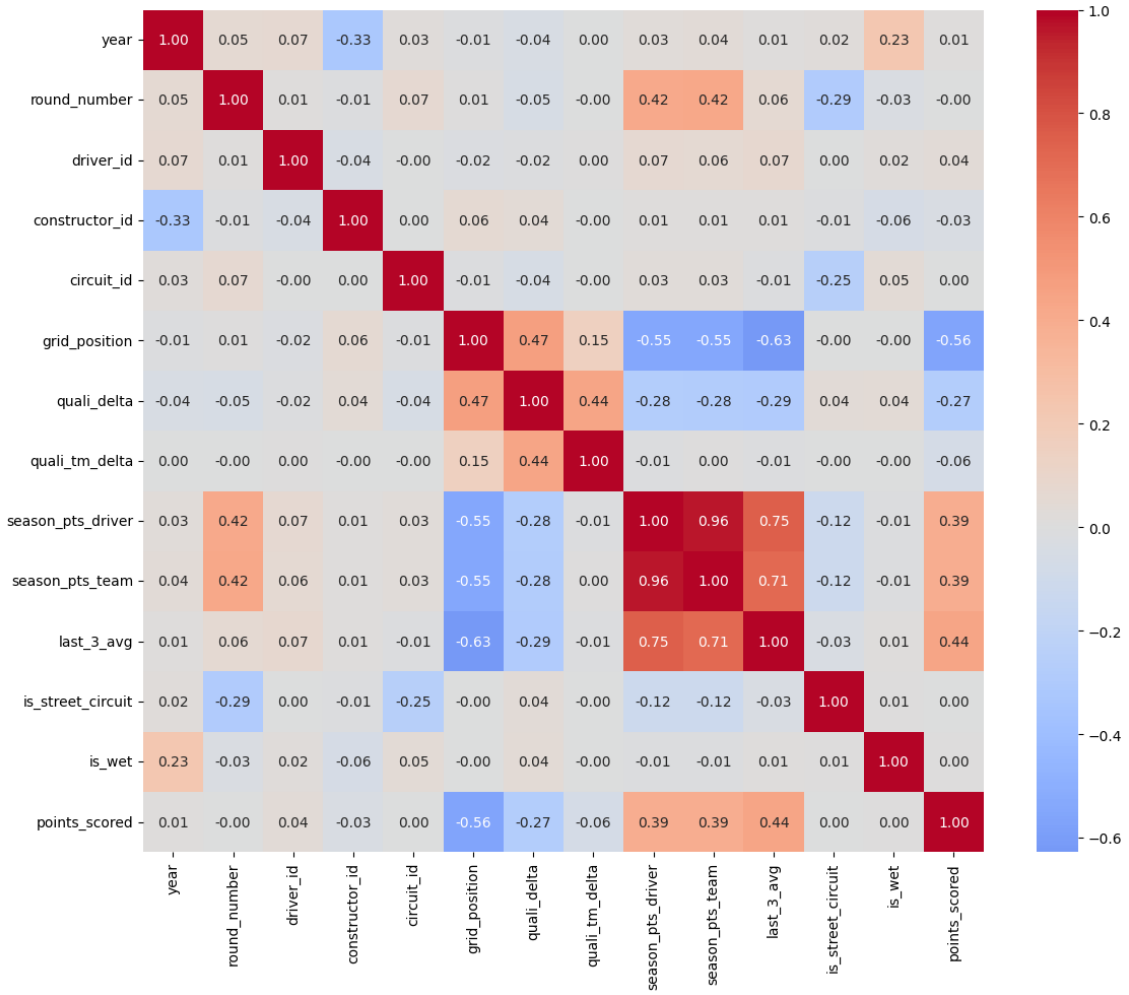

Abbildung 2: Feature Heatmap des TabPFN-Modells für die Klassifikationsaufgabe (Punktevorhersage).

4.2 Regression: Rennzeit und Gap zum Sieger

Die Notebook-Serie `notebooks/regression/` dokumentiert die Entwicklung des Regressionsziels von der absoluten Rennzeit hin zum robusteren `gap_to_winner` und nutzt dabei die in Abschnitt *Modellarchitekturen* beschriebenen Familien (Gradient Boosting/LightGBM, Random Forest, TabPFN sowie lineare Baselines). Die konkreten Experimente liegen in `baseline_regression.ipynb`, `gap_to_winner_regression.ipynb`, `random_forest_regression.ipynb` und `tabpfn_regression.ipynb`. Die Entwicklung umfasste zudem Tests mit Log-Transformationen und Normalisierungen der absoluten Zeit, welche jedoch keine signifikante Verbesserung brachten. Der Daten-Builder wurde daraufhin angepasst, um konsistente Rennzeiten (Siegerzeit + Gap) zu generieren und relevante Metriken wie `race_status`, `laps`, `gap_to_winner` und `race_time_per_lap` zu speichern.

Als neueste Erweiterung wurde ein separater Datensatz mit Trainingssessions (FP1–FP3) ergänzt (`data/regression/with_training_sessions/`).

Dieser erweitert das Feature-Set um Bestzeiten, relative Abstände zur Session-Bestzeit und Rundenanzahlen (`fp1_best`, `fp2_best`, `fp3_best`, `fp1_rel`–`fp3_rel`, `fp1_laps`–`fp3_laps`). Die zugehörigen Experimente liegen in `notebooks/regression/with_training_sessions/` und replizieren die Baselines (LGBM, RF, XGBoost, TabPFN) mit dem erweiterten Feature-Set.

5 Ergebnisse und Diskussion

5.1 Ergebnisse und Diskussion

Die Ergebnisse der durchgeführten Experimente werden in diesem Abschnitt vorgestellt und diskutiert. Die zentralen Metriken der getesteten Modelle sind in den Tabellen 5.1.1 und 5.1.2 zusammengefasst.

5.1.1 Klassifikation (Points Scored)

Für die Klassifikationsaufgabe, die Vorhersage, ob ein Fahrer Punkte erzielt, wurden mehrere Modelle verglichen. Die Ergebnisse auf dem Testset sind in Tabelle 5.1.1 dargestellt.

Modell	F1-Score (binär)	Accuracy	Balanced Accuracy
TabPFN	$\approx$ **0,855**	$\approx$ **0,850**	$\approx$ **0,850**
Random Forest	$\approx 0,835$	$\approx 0,829$	$\approx 0,829$
XGBoost	$\approx 0,763$	$\approx 0,747$	$\approx 0,747$
Neuronales Netz	$\approx 0,746$	$\approx 0,733$	$\approx 0,733$
Baseline (Logit/XGB)	$\approx 0,82$	$\approx 0,82$	$\approx 0,82$

Tabelle 1: Ergebnisse der Klassifikationsmodelle zur Vorhersage von Punkterfolgen (Testset).

5.1.2 Regression (Gap-to-Winner)

Für die Regressionsaufgabe wurde der Abstand zum Sieger (`gap_to_winner`) als Zielgröße modelliert. Tabelle 5.1.2 zeigt die wichtigsten Fehlermetriken auf dem Testset. Die ursprüngliche Modellierung der absoluten Rennzeit erwies sich mit einem MAE von über 600 s als ungeeignet, weshalb diese Ergebnisse hier nicht weiter aufgeführt werden.

Modell	MAE	RMSE
Random Forest	21,8 s	28,0 s
TabPFN	22,7 s	28,9 s
LightGBM	23,2 s	30,1 s

Tabelle 2: Ergebnisse der Regressionsmodelle zur Vorhersage des gap_to_winner (Testset).

Interpretation. Wie Tabelle 5.1.1 zeigt, erzielt das vortrainierte TabPFN-Modell die besten Ergebnisse für die Klassifikation. Dicht gefolgt wird es vom Random Forest, was die Stärke von baumbasierten Modellen für diese Art von tabellarischen Daten unterstreicht. XGBoost und das einfache neuronale Netz können auf dem Testset nicht mithalten.

In der Regression (siehe Tabelle 5.1.2) liefert der Random Forest den niedrigsten MAE, knapp vor TabPFN und LightGBM. Der Wechsel des Ziels von der absoluten Rennzeit auf den `gap_to_winner` hat die Varianz der Zielgröße signifikant reduziert und aussagekräftige Vorhersagen ermöglicht. Die verbleibenden Fehler sind vermutlich auf nicht modellierte Rennkontext-Features (z. B. Safety-Car-Phasen, Boxenstopps) zurückzuführen. Prozentuale Metriken wie SMAPE reagieren bei sehr kleinen Gaps überproportional, weshalb MAE und RMSE die primären Bewertungsmetriken bleiben.

Wichtig für die Validität der Ergebnisse ist die strikte, jahresbasierte Trennung der Daten, um eine Datenleckage durch Informationen aus der Zukunft zu vermeiden.

6 Fazit und Ausblick

6.1 Zusammenfassung und Erkenntnisse

Die Experimente bestätigen, dass eine zuverlässige Vorhersage von Rennergebnissen möglich ist. Bei der Klassifikation der Punkteplatzierungen lieferte das TabPFN-Modell mit einem F1-Score von 0,855 die besten Ergebnisse, während in der Regression der Random Forest (MAE 21,8 s) am stärksten abschnitt. Entscheidend für den Erfolg waren dabei die Nutzung des Zeitabstands zum Sieger als robuste Zielvariable sowie eine saubere Datentrennung nach Jahren. Die Genauigkeit wird derzeit jedoch noch durch die fehlende Modellierung von Ausfällen (DNFs) und chaotischen Rennverläufen eingeschränkt. Die Ausfallquote wurde mit dem Skript `calc_retirement_rate.py` berechnet und zeigte in den Jahren von 2015-2025 eine durchschnittliche Rate von etwas 16,50%. Das könnte eine mögliche Limitation sein, welche es nicht erlaubt noch höhere Genauigkeiten zu erzielen.

6.2 Ausblick

Künftige Arbeiten sollten zusätzliche Kontextdaten wie Safety-Car-Phasen und Boxenstopps integrieren, um die Realität besser abzubilden. Darüber hinaus wäre es interessant, mit Live-Daten bis zu einem definierten Zeitpunkt im Rennen zu arbeiten, etwa bis zur Rennhälfte oder bis zum letzten Boxenstopp. Dadurch könnten Zwischenstände, aktuelle Pace und strategische Entscheidungen direkt berücksichtigt werden, was die Vorhersagegenauigkeit insbesondere bei späten Rennereignissen erhöhen dürfte.

Literatur

Antaya, Mariana. (2025). *2025_f1_predictions* [GitHub repository]. Verfügbar 6. Januar 2026 unter https://github.com/mar-antaya/2025_f1_predictions

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1).

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System [arXiv:1603.02754]. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://arxiv.org/abs/1603.02754

Contributors, F. (2024). *FastF1: Formula 1 data in Python* [Python package]. https://github.com/theOehrly/Fast-F1

Hollmann, N., Müller, S., Eggensperger, K., & Hutter, F. (2022). TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second [arXiv:2207.01848]. https://arxiv.org/abs/2207.01848

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems*.

Mehta, N. (2019). Verfügbar 6. Januar 2026 unter https://medium.com/@nityachintan/how-i-built-an-f1-race-prediction-app-as-my-first-machine-learning-project-7e2e9cc89826

Roger, W. (2020). *Formula 1 Race Prediction* [Kaggle notebook]. Verfügbar 6. Januar 2026 unter https://www.kaggle.com/code/yanrogerweng/formula-1-race-prediction