Vorhersagen verschiedener Formel 1 Events

Fiete Scheel

27. Januar 2026

Inhaltsverzeichnis

1 Einleitung

1.1 Code und Daten

Der Code der in diesem Projekt verwendet wurde ist unter https://github.com/mcfiet/dl-project zu finden.

1.2 Motivation

Die Formel 1 ist ein datengetriebener Sport, in dem Performance von vielen Faktoren abhängt: Fahrer, Team, Strecke, Startposition und Rennverlauf. Durch die Verfügbarkeit strukturierter Telemetrie- und Ergebnisdaten lässt sich genau bestimmen, wie stark diese Einflüsse sind. Vorhersagen sind für Fans, Teams und Medien interessant, weil sie Vergleiche über Saisons hinweg ermöglichen und Entscheidungen (z.B. Strategieeinschätzungen) stützen können.

1.3 Problemstellung

Die zentrale Frage dieser Arbeit ist, ob sich aus vor dem Rennen verfügbaren Informationen eine zuverlässige Vorhersage über Rennergebnisse ableiten lässt. Konkret wird ein Klassifikationsproblem betrachtet: Für jeden Fahrer soll vorhergesagt werden, ob er in einem Grand Prix Punkte erzielt.

Darauf aufbauend wird ein Regressionsproblem formuliert, das den Abstand zum Sieger (`gap_to_winner`) in Sekunden modelliert. Absolute Rennzeiten zeigen sich als zu verrauscht (DNFs, rote Flaggen, Safety Cars); der Gap dient als stabileres, saisonübergreifendes Target und erlaubt perspektivisch eine Positions- bzw. Podiumsprognose. Die Herausforderung liegt in der Mischung aus kategorischen und numerischen Merkmalen, zeitlicher Dynamik sowie in der begrenzten Anzahl an Beispielen pro Saison.

1.4 Ziel dieser Arbeit

Ziel ist der Aufbau eines reproduzierbaren Datensatzes aus FastF1-Daten sowie die Entwicklung und der Vergleich geeigneter Machine-Learning-Modelle für die Punktevorhersage. Dazu werden Merkmale wie Startposition, Qualifying-Performance, Saisonschnitt der Punkte und Streckentypen abgeleitet. Die Modelle werden auf Trainings-, Validierungs- und Testdaten evaluiert und mit geeigneten Metriken (u.a. F1-Score und Balanced Accuracy) verglichen.

Ähnlich wie bei der Klassifikation, wird auch bei der Regression ein Builder entwickelt, der konsistente Rennzeiten rekonstruiert, Gaps ableitet und robuste Fehlermaße (MAE/RMSE, SMAPE) verwendet. Ziel ist eine belastbare Basis, um anschließend Podiums- oder Siegerprognosen abzuleiten. Abschließend werden die Ergebnisse interpretiert und Limitationen sowie mögliche Erweiterungen diskutiert.

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, 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`), gleitender 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.

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.

- **Gradient Boosting:** XGBoost/LightGBM als Haupt-Workhorse für tabellarische Daten; effizient in Interaktionen und nichtlinearen Effekten.
- **Neuronale Netze:** Mehrschichtige Perzeptren mit Early Stopping als tiefes Pendant, hauptsächlich für die Klassifikation erprobt.
- **TabPFN:** Vortrainiertes Few-Shot-Transformer-Modell, 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 2022, Validierung 2023, Test 2024 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. Berichtete Metriken: *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.

4 Experimente

4.1 Klassifikation: Punktevorhersage

Die Notebook-Serie `notebooks/points_scored/` dokumentiert die Experimentreihen, in denen verschiedene Modelle zur Vorhersage von Punkterfolgen trainiert und evaluiert wurden:

- **Baseline (Logit/XGB, `baseline_training.ipynb`):** Ein einfaches Baseline-Modell wurde mit One-Hot-kodierten Features trainiert.
- **Random Forest (`random_forest_baseline.ipynb`):** Ein Random-Forest-Klassifikator wurde als robustes, nichtlineares Modell eingesetzt.
- **XGBoost (`xgboost_training.ipynb`):** Ein XGBoost-Modell wurde als Vertreter der Gradient-Boosting-Verfahren evaluiert.
- **Neuronales Netz (`model_training.ipynb`):** Ein mehrschichtiges Perzeptron (MLP) mit Early Stopping wurde als Deep-Learning-Ansatz getestet.
- **Lernkurven (`learning_curve_analysis.ipynb`):** Die Abhängigkeit der Modell-Performance von der Größe des Trainingsdatensatzes wurde analysiert.
- **TabPFN (`tabpfn.ipynb`):** Ein vortrainiertes Transformer-Modell (TabPFN) wurde ohne Hyperparameter-Optimierung verwendet.

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.

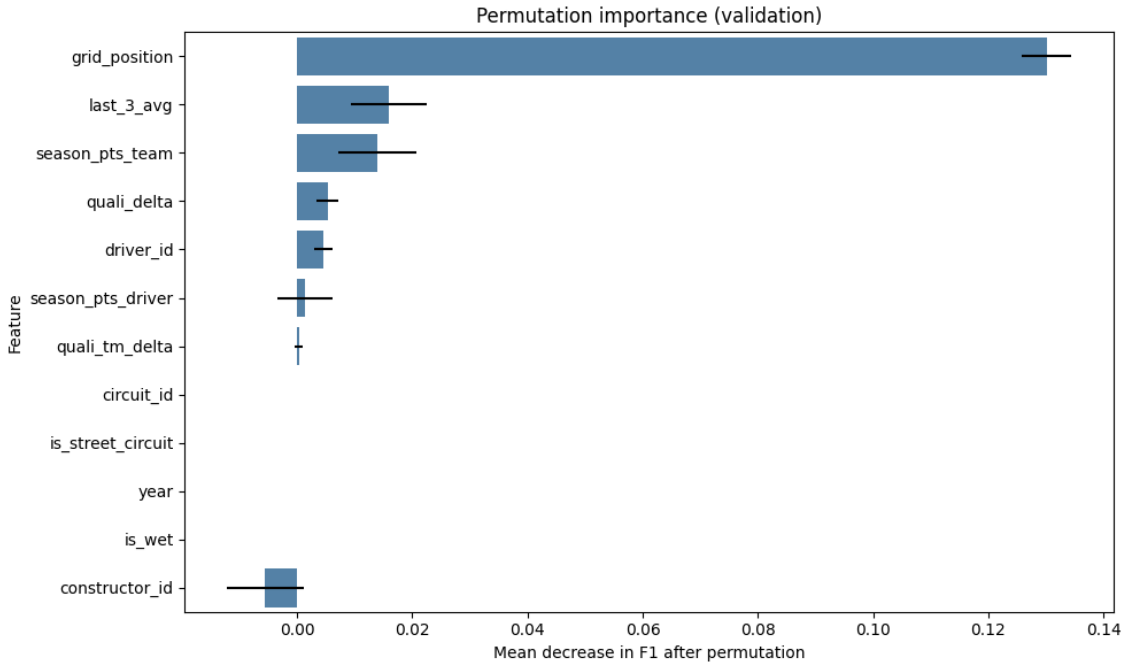

Abbildung 1: Feature Importance des TabPFN-Modells für die Klassifikationsaufgabe (Punktevorhersage).

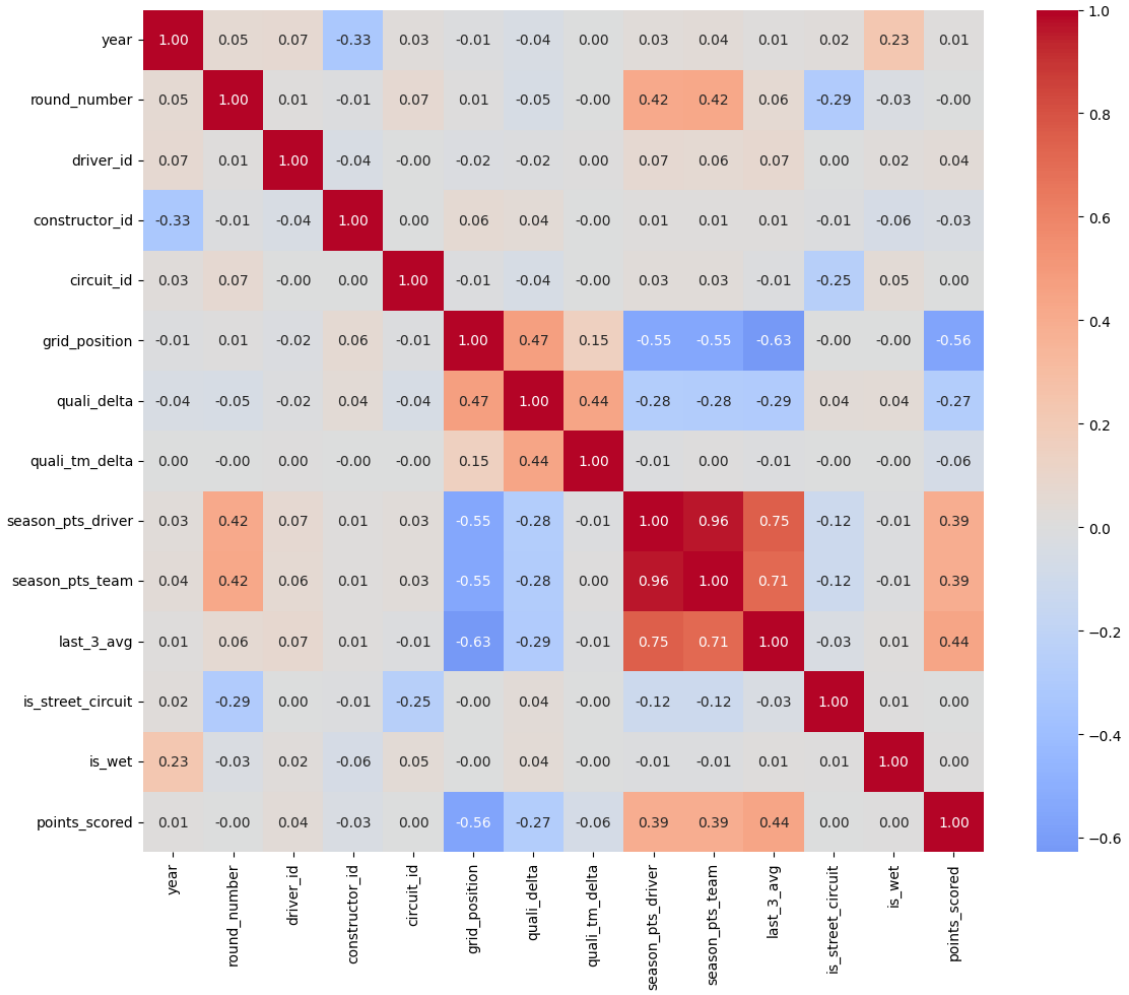

Abbildung 2: Feature Heatmap des TabPFN-Modells für die Klassifikationsaufgabe (Punktevorhersage).

4.2 Regression: Rennzeit und Gap zum Sieger

Die Notebook-Serie `notebooks/regression/` dokumentiert die Entwicklung des Regressionsziels. Ursprünglich wurde die absolute Rennzeit als Zielvariable verwendet, was sich jedoch als zu verrauscht erwies. Spätere Experimente fokussierten sich auf die Vorhersage des `gap_to_winner`:

- **Absolute Zeit (`baseline_regression.ipynb`):** Erste Versuche, die absolute Rennzeit vorherzusagen, führten zu hohen Fehlermetriken, bedingt durch Ausfälle (DNFs) und unvorhersehbare Rennereignisse.
- **Gap-to-Winner (`gap_to_winner_regression.ipynb`):** Ein LightGBM-Modell wurde trainiert, um den Abstand zum Sieger vorherzusagen, wobei die Anzahl der gefahrenen Runden (`laps`) als zusätzliches Feature diente.
- **Random Forest (`random_forest_regression.ipynb`):** Ein Random-Forest-Regressor wurde ebenfalls auf dem Gap-to-Winner-Ziel evaluiert.

- **TabPFN (`tabpfn_regression.ipynb`):** Auch für die Regression wurde TabPFN als Vergleichsmodell herangezogen.

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_session` und replizieren die Baselines (LGBM, RF, XGBoost, TabPFN) mit dem erweiterten Feature-Set.

4.3 Zusammenfassung

Für die Klassifikation wurden baumbasierte Modelle (Random Forest, XGBoost), lineare und tiefe Baselines sowie TabPFN verglichen. In der Regression erwies sich nach initialen Tests mit absoluten Rennzeiten der `gap_to_winner` als robustes Ziel. Die Grundlage für die Analyse bilden konsistente Datensätze, jahresbasierte Splits und eine klare Protokollierung der Metriken in den Notebooks.

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.

5.2 Limitationen.

- Relativ kleine Trainingsmengen pro Saison

- Keine explizite Modellierung von DNFs/Safety-Car/Boxenstopps
- TabPFN ohne systematische Tuning-Studie

Diese Punkte begrenzen derzeit die Generalisierung auf neue Saisons.

6 Fazit und Ausblick

6.1 Fazit und Ausblick

Stand der Ergebnisse. Die durchgeführten Experimente zeigen, dass eine zuverlässige Vorhersage von Rennergebnissen möglich ist. Für die Klassifikation (Punktevorhersage) lieferte das vortrainierte TabPFN-Modell mit einem F1-Score von 0,855 die beste Performance, knapp vor dem Random Forest (F1-Score 0,835). Dies unterstreicht die Stärke moderner, auf tabellarische Daten spezialisierter Architekturen. In der Regression erwies sich der Wechsel auf den `gap_to_winner` als entscheidend, um die Varianz des Ziels zu reduzieren. Hier erzielte der Random Forest mit einem MAE von 21,8 s das beste Ergebnis, dicht gefolgt von TabPFN (22,7 s) und LightGBM (23,2 s).

Kritik an der Arbeit

- DNFs und außergewöhnliche Rennverläufe werden nicht explizit modelliert, was die Vorhersagegenauigkeit bei chaotischen Rennen einschränkt.
- Die Aussagekraft von prozentualen Metriken (z.B. MAPE) bei sehr kleinen Gaps wird im Ergebnisteil zwar erwähnt, aber nicht systematisch behandelt.
- Eine automatisierte Auswertung der Experimente fehlt, was die Reproduzierbarkeit erschwert und eine Integration in CI/CD-Pipelines verhindert.

Learnings

- Jahressplits und das Maskieren von für das Testset unbekannten Kategorien sind zentral, um eine optimistische Fehleinschätzung durch Datenlecks zu vermeiden.
- Für tabellarische Daten bieten Ensemble-Methoden (Random Forest, XGBoost, LightGBM) einen starken und robusten Ausgangspunkt.
- Die Wahl und das Engineering der Zielgröße (Target, z.B. absolute Zeit vs. Gap) kann entscheidend sein, um Messrauschen zu reduzieren und ein sinnvolles Modell zu trainieren.

Ausblick

- Ergänzung von Kontext-Features wie Safety-Car-Phasen, Virtual-Safety-Car-Phasen, Boxenstopp-Daten und der Renndistanz.
- Entwicklung von zweistufigen Modellen, die zuerst die Wahrscheinlichkeit eines Rennabschlusses (Finisher-Prognose) und dann die Gap-Regression modellieren.
- Systematische Hyperparameter-Suche für die Top-Modelle (RF, XGB, LGBM) und ein erneuter, detaillierter Vergleich mit TabPFN.
- Aufbau einer automatisierten Report-Pipeline, die Metriken aus den Notebooks extrahiert und reproduzierbar als Tabellen und Grafiken für die Publikation exportiert.
- Erweiterung der Datenbasis um weitere historische Saisons oder andere Rennserien, um die Generalisierungsfähigkeit auf echten Holdout-Jahren zu validieren.

Literatur

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

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