Ergebnis Programming Contest:

Einleitung:

Hallo zusammen. In dieser PDF präsentiere/erläutere ich meine Ergebnisse für den Programming Contest von IT-Talents. An dieser Stelle möchte ich mich für die Organisation bei IT-Talents, sowie Materna bedanken und für die Möglichkeit, dass eigene Können unter Beweis zu stellen. Ziel meiner Abgabe war es den vorhandenen Datensatz zu erforschen, aufzubereiten, zu analysieren und letztendlich zu Präsentieren.
Die Präsentation meiner Ergebnisse erfolgt einerseits in diesem PDF, wo ich erläutere was mir während der Durchführung des Projektes aufgefallen ist, welche Fragen ich an den Datensatz gestellt habe, welche Erkenntnisse ich gewonnen habe und welche Ergebnisse ich aufgrund von Zeitmangel nur teilweise oder nicht bieten konnte.
Meine Abgabe hat 4 wichtige Bereiche. Dieses PDF ist der erste, ein Jupyter Notebook der Zweite, das Frontend der Dritte Und das Backend entsprechend der Vierte.
Mein Projekt ist angelehnt an das Crisp-dm best practise für Datenerforschung und geht diese Schritte grob nach.

Erforschung des Datensatzes

Um den Datensatz zu verstehen, bzw. das den Bereich, in dem der Datensatz erstellt/genutzt wird zu verstehen, musste ich erst mal eine oberflächliche Erforschung durchführen. Dafür habe ich ein Jupyter Notebook erstellt und per Import die Bibliothek Pandas eingefügt. Dann habe ich den Datensatz so geladen, wie ich ihn auf der Wettbewerbs-Website gefunden habe. Ziel war es am Anfang eine Übersicht über die Anzahl der Datensätze zu bekommen, welche Spalten es gibt, welche Typ spalten es sind, und ob bereits Auffällige Inhalte ersichtlich sind. Der rohe Datensatz hat eine Größe von 166609 Einträgen/Reihen und 12 Spalten. Diese Spalten sind : id, race_created, race_driven, track_id, challenger, opponent, money, fuel_consumption, winner, status, forecast, weather.
Inhalte der Spalten sind Bereits auf der Website von It-Talents erklärt weswegen ich an dieser Stelle nicht weiter darauf eingehen werde. Mit dem DataFrame Object „.dtypes“ habe ich die Datentypen der Zeilen schnell einsehen können, wo die ersten Ansätze für die Datenbearbeitung entstanden sind. Die 2 Spalten Race_Driven und Race_created sind vom Typ Objekt, sollten aber date oder dateTime sein. Fuel_consumption ist ebenfalls ein Objekt, sollte aber vom Typ Float sein. Status sollte statt Objekt vom Typ Integer sein, weather sollte hier statt Objekt String sein. Der Forecast ist ebenfalls ein Objekt, was als javascript Object oder Python Dictionary okay wäre, da es Key Value Pairs beinhaltet. Um eine Analyse zu ermöglichen, muss hier auch eine Konvertierung erfolgen. Im nächsten Schritt habe ich pandasql installiert da ich gerne mit SQL arbeite und mit dem Paket SQL-Abfragen an Pandas Dataframes stellen kann. Hier habe ich für das Allgemeine Verständnis Queries geschrieben, diese beinhalteten eine Filterung nach meiste Siege, Meiste Verluste, höchste und Niedrigste Daten einer Spalte, ob Spalten none oder im Pandas als NaN angegeben sind. Hier ist mir ebenfalls aufgefallen, dass in der Spalte fuel_consumption statt Float werte ebenfalls Datumswerte angegeben sind, was in der Datenvorbereitung ebenfalls entfernt werden sollte.
Das Business bewegt sich im Umfeld eines Rennspiels. Hier können Nutzer andere zu einem Rennen Herausfordern. Dieses erfolgt auf einer spezifischen Strecke mit einem festgelegten Gewinneinsatz. Der Gewinner erhält das Geld. Wird eine Herausforderung gestellt, wird ein

Zeitstempel der Tabelle hinzugefügt in der Spalte Race_Created. Wird das Rennen auch gefahren, wird zum Zeitpunkt der Austragung ein Zeitstempel in die Spalte Race_Driven hinzugefügt. Eine Wetter Prognose wird für den Tag erstellt und das tatsächliche Wetter wird ebenfalls erfasst, sofern das Rennen ausgetragen wurde.

Aufbereitung des Datensatzes

Nach der Erforschung des Datensatzes geht es um die Datenvorbereitung. Ziel hier ist es die Daten so aufzubereiten, dass Fehlerhafte Spalten entfernt werden, fehlerhafte Wertetypen konvertiert werden, und zur Analyse ungeeignete Typen umgeformt werden. Die Aufbereitung ist einsehbar in der Datei Data_Preprocessing.ipynb im Ordner Preprocessing. An dieser werde ich mich entlang arbeiten.

Konvertierung vom Typ Object und Löschung von NaNs:

Im Ersten Schritt konvertiere ich alle Datentypen, wo es ohne Umwege möglich ist. Das wären hier race_created, race_driven und fuel_consumption. Die Konvertierung der fuel_consumption von Object zu Float hat einen positiven Nebeneffekt, dass alle Datumswerte in NaNs, also nicht angegebene Werte, umformatiert werden. Diese können dann ohne Probleme im nächsten Schritt per Pandas Funktion .dropna() entfernt werden. Die beiden Datums-Felder sind vom unterschiedlichen Typ. Race_created ist nur als Date angegeben, race_driven wiederum hat auch eine Zeit und ist dementsprechend zu dateTime zu konvertieren. Der Einfachheit halber zum gegenseitigen Filtern im späteren Verlauf habe ich beide Felder zu dateTime umformatiert. Alle ungültigen Formate werden auch hier in NaNs umgewandelt und mit dropna entfernt.

Codieren von nominalen Werten für spätere Analye (z.B. Machine Learning)

Die Spalten weather und status sind vom Typ Objekt und könnten in Strings umgewandelt werden. Um den Datensatz aber auch für Machine Learning vorzubereiten codiere ich diese Werte direkt in Integer. Dafür Nutze ich die Funktion LabelEncoder von sklearn Preprocessing, welche eine einfache Codierung und Decodierung ermöglicht. Damit werden alle möglichen Werte, welche die Spalte annehmen kann, erfasst und zugehörige Wertepaare gebildet. Z.B. wird gelb, grün in 0,1 umgewandelt. Mit dem Dekodierer kann man diese Werte in ihre Ursprüngliche Form umwandeln. Die Dekodierung ist notwendig, da nominale Felder die Performance von Machine Learning Algorithmen negativ beeinflussen kann und dem möchte ich Vorbeugen.

Löschung von nicht-gefahrenen-Rennen.

In diesem Schritt lösche ich alle Rennen, welche nicht ausgetragen wurden. Für mich stand Aussagekraft von allen Rennen die tatsächlich gefahren wurden in diesem Projekt im Fokus. Auch wenn nicht vorhandene Daten, oder unvollständige Daten ebenfalls eine Aussagekraft haben, und ich auch zu diesen Fragestellungen setzen und erarbeiten könnte, wollte ich mich im Scope dieses Projektes nur auf gefahrene Rennen spezialisieren. Mit mehr Zeit hätte ich auch die nicht gefahrenen Rennen analysiert. Nach der Dekodierung hatten ausgetragene Rennen den Status 1, nach diesem habe ich gefiltert und alles gelöscht was nicht 1 als Status hatte.

Löschung von Wett-Einsätzen größer 1.000.000

Dies war eine Vorgabe, bzw. ein Hinweis vom Auftraggeber, dass Werte in dieser Spalte zwischen 30 und 1.000.000 liegen, wenn es gültige Rennen sind. Auch hier habe ich die spalte money gefiltert nach größer 1.000.000 oder kleiner 30 und diese Einträge gelöscht.

Löschung von Rennen, wo die Austragung vor der Anmeldung war

In der Spaltenbeschreibung wurde angegeben, dass ein Rennen erst angemeldet werden muss, bevor es ausgetragen werden kann. Nach dieser Bedingung habe ich alle Rennen als Fehlerhaft aufgenommen deklariert, welches den Zeitpunkt der Austragung vor dem Zeitpunkt der Anmeldung hatten. Hier anzumerken ist, dass ich Rennen nur entfernt habe, wenn es mindestens einen Tag vorher ausgetragen wurde. Filterung am selben Tag, aber früherer Austragungszeit war nicht möglich, da race_created keine Zeitangabe enthält. Auch diese Einträge habe ich gelöscht.

Hinzufügen von Wetter-Prognose Spalten und Iterieren über Wetter Prognose

Auch die Wetter Prognose wollte ich für mögliche Machine learning Algorithmen vorbereiten und hier nur Integers in der jeweiligen Spalte zu haben. Um dies zu ermöglichen habe ich 4 Spalten hinzugefügt, welche die möglichen Arten des Wetters beinhalten (Sonne, Regen, Schnee, Gewitter). In den Spalten selbst steht als integer der Prozentuale Wert * 100 zur dazugehörigen Wetterprognose. Um die Werte zu bekommen habe ich mit Regular Expressions den forecast String gefiltert. Hier habe ich mit dem Ausdruck „'i:(\d*)'“ alle Werte im String der forecast Spalte extrahiert und einer Liste gespeichert. Die einzelnen Werte weise ich den dazugehörigen Spalten zu und lösche danach die ursprüngliche Spalte forecast. Zum Abschluss konvertier ich die Werte zum Typ integer.
Anmerkung: Die Laufzeit der DataFrame Iteration, Anwendung von Regex und Zuweisung der Tabelleninhalte ist problematisch. Da ich Laufzeitoptimierung Zeittechnisch nicht mehr geschafft habe dauert dieser Schritt länger. Mit mehr Zeit wäre hier noch eine Optimierung möglich gewesen, ich habe meine erste Möglichkeit, die funktioniert hat, verwendet.

Führt man diese Datenbereinigung durch, bleiben etwa 70000 Datensätze übrig, die zur weiteren Auswertung genutzt werden und lokal in einer CSV gespeichert werden. Es wären mehr Datensätze übriggeblieben, wenn ich auch nicht ausgetragene Rennen zur Analyse genutzt hätte, aber im Bereich meines Projektes war das nicht vorgesehen.

Fragen an den Datensatz

Meine Fragen an den Datensatz können in 2 Bereiche eingeteilt werden: Vergangenheit und Zukunft.
In der Vergangenheit habe ich mir Fragen zum Thema Analyse und Visualisierung gestellt. Dazu gehören

1. Wie viele Rennen hat ein Spieler Gewonnen/Verloren? Welche Gewinnrate hat er?
2. Wie viel Geld hat ein Spieler Verloren/Gewonnen?
3. Hat ein Spieler Favorisierte Strecken? Wenn ja, Welche?
4. Hat ein Spieler Erzfeinde? Also Fahrer, gegen die er besonders oft angetreten ist?

5. Welches waren die letzten angetretenen Rennen eines Spielers? Wie sind diese ausgefallen?
6. Welche Rekorde konnten aufgestellt werden? Rekorde wie meiste Rennen, meistes Geld Gewonnen, welche Strecke wurde am meisten Befahren, meistes Geld verloren.

Diese Fragen sind statistisch nicht ordentlich formuliert, waren aber mein Ansatz bei der Erstellung des Ergebnisses.

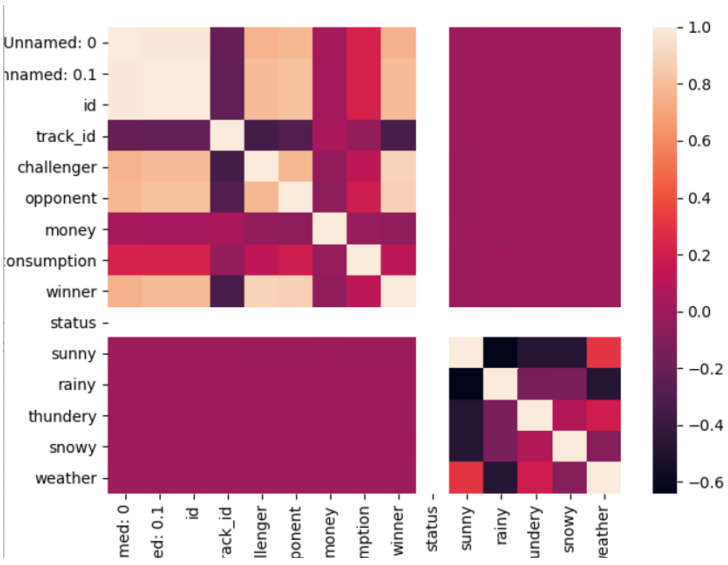
In der Zukunft geht es um Prognosen, also Vorhersagen, auf Basis der vergangenen Daten, um Rennergebnisse für die Zukunft zu erhalten. Hier haben mich folgende Fragen interessiert

1. Gibt es Korrelationen (z.B. nach Pearson) zwischen den Spalten? Hier spezifisch, kann man einen Sieg vorhersagen, je nachdem welche Strecke Gefahren wurde, welches Wetter es ist, wer Herausforderer/Gegner ist oder welcher Sprit verbraucht wurde.
2. Kann man ein Machine Learning Modell auf dem Datensatz trainieren, welches zur Vorhersage der Rennergebnisse verwendet werden kann?

Ergebnisse-Zukunft

Zukunft:
Da die Ergebnisse und Anzahl der Fragen zur Zukunft geringer sind starte ich mit diesem Bereich.

Eine schnelle Übersicht zu Korrelationen mittels Seaborn und Pandas
ergibt folgende Heatmap:

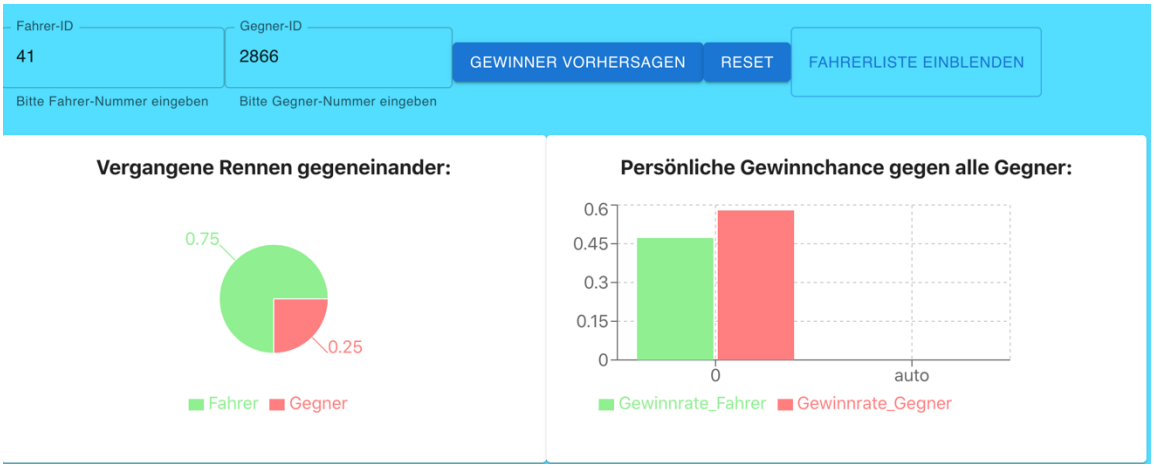


Hier ist zu sehen, dass das Wetter keinen signifikanten Einfluss auf das Rennergebnis hat. Auch die fuel Consumption hat zu Winner nur eine geringe Korrelation. Hohe Korrelationen zwischen opponent, challenger und money ist zu verfolgen, was ich statistisch aus Zeitmangel nicht mehr durchgeführt habe.

Auch zum Opfer von Zeitmangel ist die Probe eines Machine learning Models gefallen. Hier wollte ich mit der Python Bibliothek sklearn eine Machine Learning bzw. Neuronales Net-

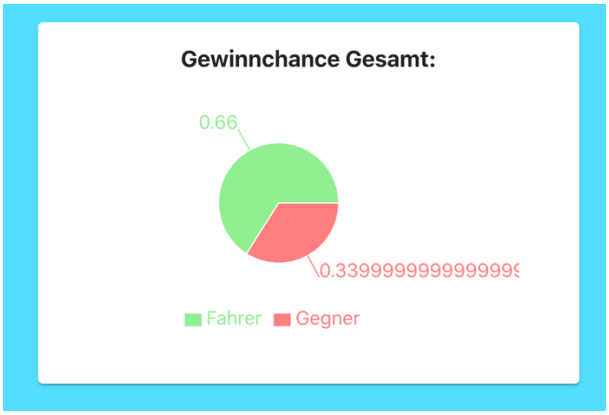
Lösung für die Vorhersage von Rennergebnissen ausprobieren. Wie man dem Frontend entnehmen kann, habe ich eine Prognose Seite erstellt, wo man die Chance ermitteln kann, ein Rennen zu gewinnen in Abhängigkeit des Gegners. Die Siegeschance wird mit einem sehr simplen neuronalen Netz berechnet, welches 2 Werte in die Auswertung einbezieht. Der erste ist die Gewinnrate der beiden Fahrer aus der Vergangenheit, wenn sie Bereits gegeneinander gefahren sind. Der zweite Wert ist die allgemeine Persönliche Gewinnchance, die jeder der beiden Fahrer hat. Die Aktivierungsfunktion des zweiten Wertes ist eine einfache Abfrage, ob mindestens 10 Fahrten getätigt wurden, um eine Aussage treffen zu können. Hat ein Fahrer z.B. nur ein Rennen gefahren und das gewonnen, wäre die Gewinnwahrscheinlichkeit 100% und würde das Gesamtergebnis verzerren. Die Gewichtung der beiden habe ich nach eigenem Ermessen so gesetzt, dass die persönliche Gewinnrate zu 30% eingerechnet wird, und die bereits gefahrenen Rennen gegen den spezifischen Gegner zu 70%. Diese könnten auch noch angepasst werden, waren für mich aber so weit logisch.

Ein kurzer Einblick wie die Vorhersage aufgebaut ist, sieht man hier:
Fahrer ist in Grün, Gegner in Rot dargestellt.



Links: Wie viel Prozent der Rennen gegen den spezifischen Gegner gewonnen wurden (wurden keine Gefahren ist diese 50/50)
Rechts: Wie die persönliche Gewinnrate bei allen gefahrenen Rennen war.

Und die Gewinnwahrscheinlichkeit bei einem Rennen gegeneinander wird hier gezeigt.
Grün: Fahrer, Rot: Gegner



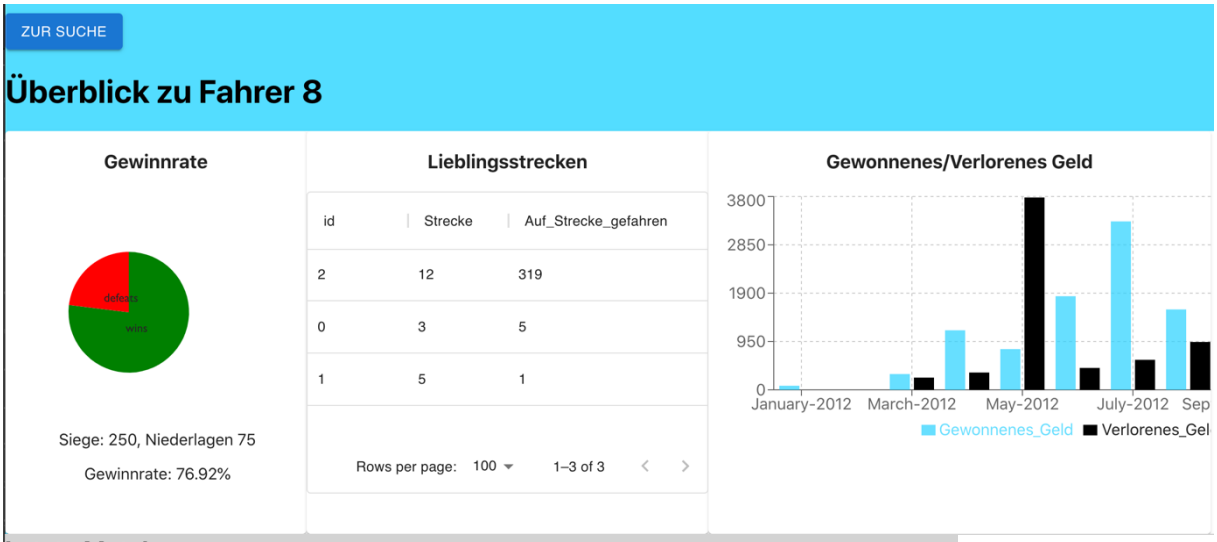
Hier zu sehen ist, dass obwohl Grün in der Vergangenheit sehr gut gegen Rot performt hat, die Gewinnrate für Rot leicht steigt, da die persönliche Gewinnrate von Rot höher ist als von Grün.

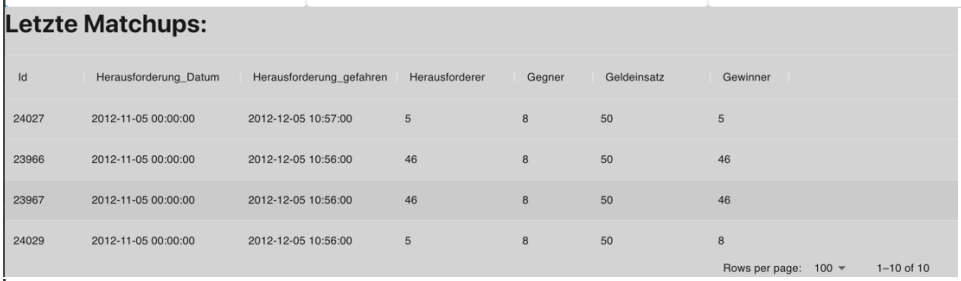
Ebenfalls einfließen bei der Berechnung sollte noch die Gewinnwahrscheinlichkeit bei bestimmtem Wetter. Hierfür würde man in der Suche neben den Gegner noch ein bestimmtes Wetter auswählen (Sonne, Regen, Schnee, Gewitter) und die Gewinnrate der eigenen Fahrten bei bestimmtem Wetter würde berechnet werden. Außerdem sollte noch die Gewinnrate auf einer spezifischen Strecke ausgerechnet werden, welche ebenfalls bei der Suche ausgewählt wird. Beides konnte ich aufgrund von Zeitproblemen nicht mehr implementieren.

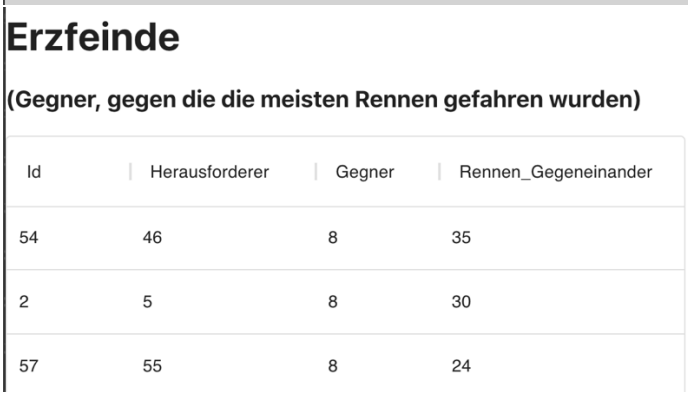
Ergebnisse-Vergangenheit

Für die Vergangenheit konnte ich viele meiner Fragen beantworten. Diese habe ich auf der Spielerprofil Seite visualisiert. Ziel hier ist es, zu einem bestimmten Spieler eine kurze statistische Übersicht zu bieten, die Wichtige Daten auf einen Blick Zeigen.

Hier ein Einblick des Profils zum Fahrer mit der Nummer 8:



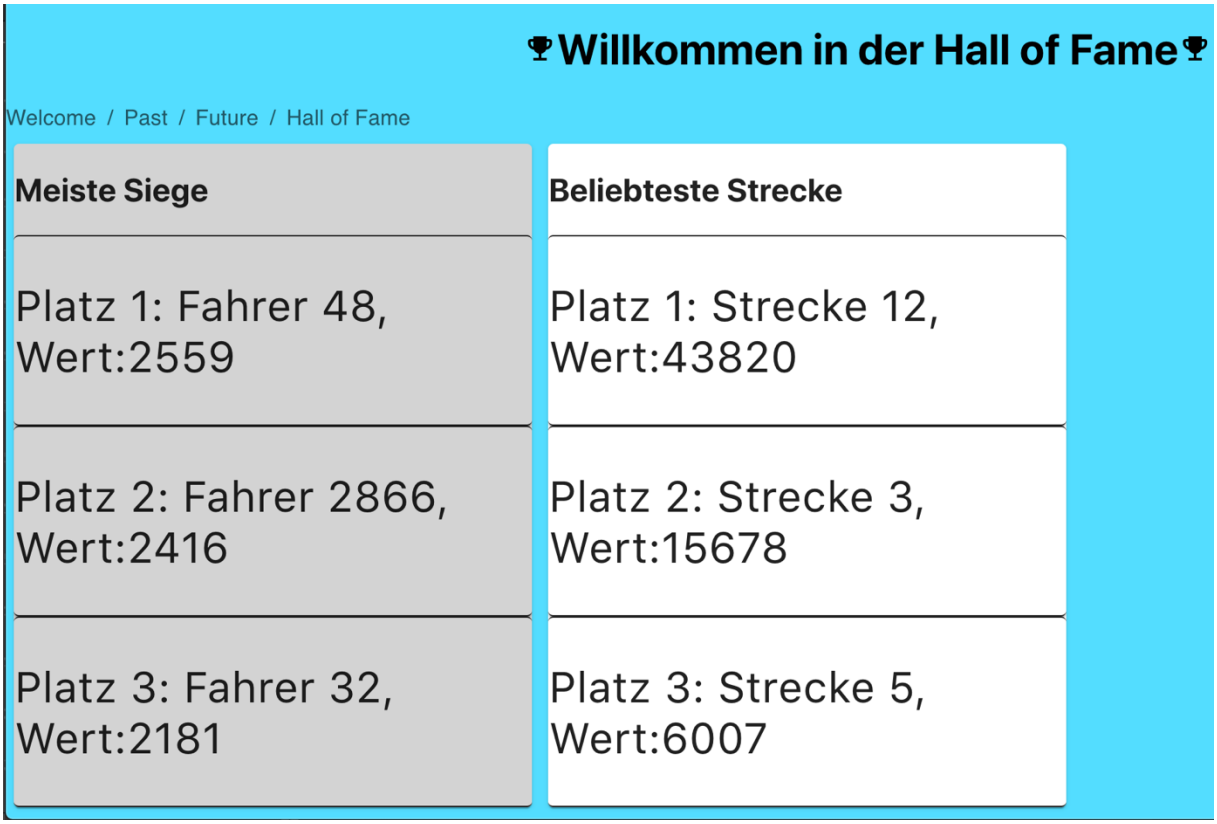




Die Gewinnrate zeigt die Performance eines Fahrers an. Je mehr Grün zu sehen ist, desto besser. Darunter befinden sich die dazugehörigen nackten Zahlen und die prozentuale Gewinnrate.
Die Lieblingsstrecke ist eine Tabelle, welche angibt auf welchen Strecken die meisten Rennen gefahren wurden. Ob es jetzt seine Lieblingsstrecke ist, konnte ich natürlich nicht nachfragen, aber auf jeden Fall hat er dort viel Erfahrung.
Das gewonnene/verlorene Geld zeigt, wie die finanzielle Lage aussieht. Das Geld wurde nach Monat geclustert und aufsummiert.
Die letzten Matchups ist eine tabellarische Übersicht der zuletzt gefahrenen Rennen und deren Ergebnisse.
Der Erzfeinde Bereich ist ebenfalls eine Tabelle, welche zeigt, gegen welchen Gegner am meisten gefahren wurde. Auch hier muss es nicht unbedingt ein Feind sein, aber gegeneinander Gefahren wurde hier am meisten.

Nicht mehr in die Übersicht geschafft hat es eine Übersicht über das Wetter, bei welchem Wetter jemand gerne oder oft fährt.

Ebenfalls zur Vergangenheit gehört die Hall of Fame. Ein Bereich wo Rekorde aus dem Datensatz dargestellt werden:



Hier sind 2 Rekorde zu sehen. Der erste huldigt den Fahrer, welcher Absolut die meisten Siege hat. Der 2. Krönt die meist befahrenste Strecke und gibt an, wie oft auf dieser gefahren wurde.
Auf Grund von Zeitproblemen sind das die einzigen beiden Rekorde, die dargestellt wurden. Ebenfalls geplant, aber nicht mehr implementiert waren:

-Am meisten Geld gewonnen/verloren
-Am meisten Rennen verloren
-Beste Gewinnrate
-Am meisten Sprit verbraucht

Die Analyse und Beschaffung der Daten sind meinem Backend Python Code zu entnehmen. Da das meiste nur Pandas Data Frame Slicing, Gruppierung und Filter waren, gehe ich hier auf die Technik nicht genauer ein.

Das Frontend

Das Frontend visualisiert meine Daten und damit auch meine Ergebnisse. Ich habe mich für eine Website entschieden, da ich mich mit der Programmierung von Applikationen (noch) nicht auskenne und ich eine Datenpräsentation in der Konsole für nicht ausreichend empfinde. Die Website wird mit der Javascript Bibliothek „React“ aufgebaut, welche einen nativen Programmierstil bietet.
Die verschiedenen Funktionen meiner Website habe ich auf unterschiedlichen Seiten ausgelagert, welche über den React-Router erreicht werden. Auf den Seiten ist teilweise eine Usereingabe erforderlich, welche eine Anfrage zum Backend Server startet. Wichtig hier ist anzumerken, dass ich ein Error-Handling nur im minimalen Stil implementiert habe. Beim Spielerprofil z.B. benötigt eine Anfrage eine spezifische Zahl. Wird ein String übergeben oder eine falsche Zahl, so wird die Anfrage an den Server abgebrochen und eine Fehlermeldung wird ausgegeben. Dieses Error handling konnte ich nicht mehr in der Prognose implementieren. Deswegen sollten für eine korrekte Darstellung der Ergebnisse nur Fahrer-Nummern eingegeben werden, welche auch im Datensatz enthalten sind. Abgesehen davon habe ich versucht, die Seite möglichst einfach zu gestalten, damit der User schnell die richtigen Funktionen findet und anwenden kann.
Der Website Code ist aus Zeitgründen nicht optimiert und man sieht die erste Fassung, die funktioniert hat. Einige Variablen sind überflüssig, importierte Pakete ungenutzt oder Laufzeit-Suboptimale Lösungen implementiert. Z.B: hätte ich gerne noch ein Ladebalken angezeigt anstatt leerer Graphen. Neben der Optimierung ist ebenfalls die Code Struktur noch sehr unübersichtlich. Viele Komponenten hätten noch in eine eigene Datei ausgelagert werden können, um doppelten Code zu vermeiden. Abgesehen davon bin ich zufrieden mit meinem Ergebnis und denke, dass die Website informativ ist, einfach zu Bedienen und einladend wirkt.

Das Backend

Das Backend verwaltet die Datenverarbeitung der Website. An dieses werden GET und POST Anfragen gestellt, um Daten zu verlangen. Die Datengrundlage wird beim Serverstart aus den aufbereiteten Daten geladen und für die weitere Bearbeitung genutzt. Als Backendserver nutze ich ein Python Datei mit der Flask Bibliothek. Python nutze ich, weil ich gerne damit arbeite und es sehr gut für die Arbeit mit Daten geeignet ist. Die Flask-Bibliothek ermöglicht es mir die Python Datei als Server lokal zu hosten und Anfrage über http-Requests an diesen zu senden. Input und Output sind JSON Formate. Über GET Anfragen werden keine Parameter mitgegeben und bei den dazugehörigen Routen wird ein

Ergebnis ohne Parameter-Abhängigkeit zurückgegeben. Bei den POST Anfragen erwarten die Paths/Funktionen die Fahrer ID, um die zu einem Fahrer zugehörigen Daten auszugeben.
Das Backend ist sehr übersichtlich aufgebaut, da jede Art von Return einen eigenen Pfad und eine eigene Funktion bekommen hat. Innerhalb mancher Abfragen sind ebenfalls suboptimale Laufzeit Lösungen implementiert, welche aus Zeitgründen nicht weiter verbessert werden (z.B. GetPrediction).
Im Backend befinden sich aufbereitete Daten, welche ich im vorherigen Schritt durch die Aufbereitungs-pipeline erstellt habe. (Siehe Jupyter Notebook Datei) Die Aufbereitung selbst wollte ich nicht im Backend laufen lassen, da sie eine längere Laufzeit hat und ich keine Verzögerung bei der Präsentation der Ergebnisse haben möchte.

Persönliche Entwicklung und Fazit

Zeitproblem ist in diesem Teil großgeschrieben. Ich konnte aufgrund von anderen Terminen leider nicht die gesamte zur Verfügung stehenden Zeit nutzen. Dadurch ist der Code in vielen Bereichen unausgereift, nicht effizient und unübersichtlich. Ergebnisse mussten gestrichen werden, und interessante Ansätze konnten nicht komplett verfolgt werden. Mein Ergebnis zeigt was möglich ist, in welche Richtung ich gehe bei der Datenanalyse sowie der Aufbereitung. Für die Zeit, die ich investiert habe, bin ich sehr zufrieden mit der Website und der Datenanalyse. Sollte in Zukunft mehr Zeit zur Verfügung stehen, kann ich bei ähnlichen Projekten optimierte Ergebnisse erstellen. Es hat mir viel Spaß gemacht durch die Daten zu gehen, eine Website zu gestalten und zu befüllen und ich denke, dass ich den Code nach der Competition noch ein wenig anpassen werde und auf meinem Github Profil präsentiere. Schade finde ich, dass ich nicht die Möglichkeit hatte mich in Machine Learning und neuronalen Netzen hier auszuprobieren. Das werde ich privat außerhalb vom Wettbewerb nachholen. Da ich im Bereich Data Science/Data Engineering/BigData arbeiten möchte war ich sehr zufrieden mit dem Thema dieses Wettbewerbs und würde mich freuen, wenn es ähnliche in Zukunft geben wird.