Raport z projektu Jakub Zaręba

F1 Lap Time Prediction

06-DUMALI0 (2024/SZ)

Cel projektu

Celem projektu było stworzenie modelu, który przewiduje czas danego okrążenia w wyścigu F1 na podstawie danych ogólnodostępnych dla widowni (bez telemetrii bolidu). Wybrany przeze mnie problem to problem regresji.

Dane

Dane pochodzą z witryny Kaggle (link: https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020/). Jest to zbiór danych zawierający m.in. praktycznie wszystkie okrążenia przejechane w F1 w latach 1950-2024. Zawiera również masę innych danych odnośnie królowej motorsportu takich jak rezultaty każdego wyścigu, informację na temat zjazdów do alei serwisowej. Dokonałem ekstrakcji potrzebnych mi cech do tego projektu, jak i pewnych przekształceń danych, by otrzymać nowe cechy. Po odrzuceniu niepełnych danych, obserwacji odstających uzyskano 464089 przykładów. Dokonałem podziału na zbiory uczące oraz testowe. Rozmiar zbioru testowego to 20% całego zbioru.

Wykorzystane cechy to:

- Identyfikatory
  - Wyścigu
  - Toru
  - Kierowcy
  - Konstruktora
- Pozycja startowa do wyścigu
- Numer obecnego okrążenia
- Obecna pozycja kierowcy na torze
- Czy obecne okrążenie jest okrążeniem zjazdowym do alei serwisowej

- Czy obecne okrążenie jest okrążeniem wyjazdowym z alei serwisowej

Informacja odnośnie wjazdu/wyjazdu z alei serwisowej jest szczególnie ważna do predykcji czasu okrążenia, ponieważ zawsze wpływa na czas okrążenia. Okrążenie zjazdowe sugeruje mocno zużyte opony lub uszkodzenia w bolidzie, wyjazdowe natomiast będzie zawsze gorsze (dłuższe) pod względem czasu z powodu niedogrzanych opon.

- Czas najlepszego okrążenia uzyskanego przez kierowcę w kwalifikacjach do danego wyścigu

Sugeruje to maksymalne tempo na które było stać kierowcę podczas danego weekendu wyścigowego.

Modele

W projekcie porównano działanie 6 modeli:

Dla niektórych modeli zostało użyte skalowanie za pomocą StandardScalera z biblioteki scikit-learn dla kolumn związanych z czasem okrążenia (kolumny: „bestQualiTime” oraz „time”). W przypadku użycia w opisie modeli będę pisał tylko „z zastosowaniem skalowania”.

- Regresja liniowa z biblioteki scikit-learn.
- Ta sama implementacja regresji liniowej, ale z zastosowaniem skalowania.
- Regresja z regularyzacją L2 (Ridge z bib. Scikit-learn). Z parametrem lambda równym 3.3, wartość została dobrana czysto „ręcznie”.
- Regresja z regularyzacją L1 (Lasso z bib. Scikit-learn). Z parametrem lambda równym 0.1, wartość została dobrana również czysto „ręcznie”.
- Model lasu losowego dla problemu regresji (RandomForestRegressor z bib. Scikit-learn). Liczba drzew została ustawiona na 50, maksymalna głębokość drzewa jest równa 30. Domyślnie kryterium błędu drzewa z której korzysta algorytm jest MSE.
- Model sieci neuronowej w PyTorch’u

Budowa sieci:

- Warstwa dropout z prawdopod. 0.3
- Trzy warstwy liniowe:
    - Pierwsza o wymiarach: rozmiar danych wejściowych na wejściu tej warstwy tj. 10, oraz 64 na wyjściu
    - Druga o wymiarach: 64 na wejściu, 32 na wyjściu
    - Trzecia o wymiarach: 32 na wejściu, 1 na wyjściu

Funkcje aktywacji warstw liniowych to *ReLU*.
Wykorzystywany optymalizator to Adam.
Funkcją błędu jest RMSE.

Rozmiar batcha podczas uczenia to 64, współczynnik uczenia to 0.001.

Model uczy się przez 20 epok.

Ewaluacja

Do ewaluacji modeli wykorzystano metryki *RMSE*, oraz MAE. Wyniki ewaluacji znajdują się w poniższej tabeli. Wartości metryk zostały zaokrąglone do 5 miejsca po przecinku.

Model	RMSE	MAE
Regresja liniowa	0.14101	0.04631
Regresja liniowa ze skalowaniem	0.62261	0.20448
Regresja liniowa z regularyzacją L2 (Ridge). lambda=3.3	0.14101	0.04632
Regresja liniowa z regularyzacją L1 (Lasso). Lambda=0.1	0.21978	0.15713
Model lasu losowego. Liczba drzew=50, maksymalna głębokość=30	0.07322	0.00794
Model sieci neuronowej	0.2921	0.2434

Wnioski

Najlepsze wyniki pod względem zarówno metryki RMSE jak i MAE osiągnął model lasu losowego. Najsłabszym modelem pod względem obu metryk okazał się model regresji wraz ze skalowaniem, być może wynika to ze względu na to, że dane były maksymalnie z przedziału ok. 0.8 minuty do 2 minut, przez co zwykłe przeskalowanie przy użyciu StandardScaler było po prostu błędem. Model sieci neuronowej również nie jest zbyt dobry w porównaniu do innych prostszych metod, być może należało zaprojektować bardziej rozbudowany model. Model lasu losowego daje dosyć dobre wyniki, co prawdopodobnie wskazuje na brak szczególnej liniowości danych do tego celu. Wydaję mi się, że po prostu cel projektu jest trochę nietrafiony pod względem dostępnych danych. Chciałem również pozbyć się danych kategorycznych, czyli kolumn z „Id" w nazwie, ale wykorzystywanie do tego techniki OneHotEncodingu powodowało wyrzucanie pythonowego kernela, pewnie z powodu ilości nowopowstałych kolumn.

Przemyślenia

Z wykorzystanych w tym projekcie danych prawdopodobnie lepszym pomysłem byłoby podejście do problemu klasyfikacji związanego z predykcją miejsca na mecie/na końcu danego okrążenia. Jestem fanem Formuły 1 i rzetelne szacowanie czasu okrążenia na podstawie takich stosunkowo podstawowych danych było dla mnie ciekawym pomysłem, mogłoby to przydać mi się przy oglądaniu wyścigów, ponieważ mógłbym w czasie rzeczywistym wiedzieć co dzieje się z każdym z tych kierowców. Zdecydowanie taki projekt ma sens przy większej ilości danych jak np. telemetria bolidu (która niestety jest nie do zdobycia publicznie, a byłoby to niesamowite pole do popisu w dziedzinie uczenia maszynowego) lub aktualne warunki atmosferyczne na torze.