Spiegazione Semplice di Fitness-to-Drive

Versione facile da leggere

21 febbraio 2026

1 Idea super semplice

Immagina un copilota digitale. Questo copilota guarda cosa succede mentre guidi e prova a rispondere a una domanda:

"Nel prossimo secondo, quanto e' probabile fare un errore?"

Il file `ftd_predictor.py` serve proprio a costruire questo copilota.

2 Che cosa significa Fitness-to-Drive

Fitness-to-Drive vuol dire: *quanto sei in forma per guidare adesso.*

Nel codice si calcola cosi:

```
fitness_to_drive = 1 - error_probability
```

Esempio:

- se `error_probability = 0.20`, allora `fitness_to_drive = 0.80`.
- se `error_probability = 0.70`, allora `fitness_to_drive = 0.30`.

Quindi:

- errore probabile basso = guida piu sicura;
- errore probabile alto = guida piu rischiosa.

3 Quali dati usa

Il programma legge 4 file CSV:

1. quando ci sono state distrazioni;
2. quando ci sono stati errori durante quelle prove;
3. errori in guida normale (baseline);
4. quanti secondi di guida normale ci sono stati.

Detto semplice: guarda **quando ti distrai**, **quando sbagli** e **come va quando non ci sono distrazioni**.

4 Le 7 feature del modello (spiegate semplice)

Il modello usa 7 "indizi" (feature). Ogni indizio e' un numeretto che aiuta a capire il rischio.

1. `distraction_active`
   Dice se in quel momento c'e una distrazione attiva: 1 = si, 0 = no.
   Arriva dalle finestre tempo in `Dataset Distractions_distraction.csv` (colonne `timestamp_start`, `timestamp_end`).

2. `time_since_last_dist`
   Dice da quanti secondi e' finita l'ultima distrazione.
   Anche questo viene calcolato usando i tempi di `Dataset Distractions_distraction.csv`.

3. `model_prob`
   E' la fiducia del modello di arousal.
   Nei campioni con errore arriva da `Dataset Errors_distraction.csv` (colonna `model_prob`).
   Nei negativi da distrazione arriva da `model_prob_start` o `model_prob_end` in `Dataset Distractions_dist`
   Nei negativi baseline usa un valore "tipico" (mediana).

4. `model_pred_enc`
   E' l'etichetta arousal trasformata in numero (perche il modello capisce meglio i numeri).
   Parte da `model_pred`, `model_pred_start`, `model_pred_end`.

5. `emotion_prob`
   E' la fiducia del classificatore emozioni.
   Parte da `emotion_prob` (errori) oppure `emotion_prob_start/end` (distrazioni), e baseline tipico.

6. `emotion_label_enc`
   E' l'emozione trasformata in numero.
   Parte da `emotion_label`, `emotion_label_start`, `emotion_label_end`.

7. `baseline_error_rate`
   E' il "rischio normale" della persona quando guida senza distrazioni speciali.
   Si calcola con:

   - `Dataset Errors_baseline.csv` (quanti errori baseline);
   - `Dataset Driving Time_baseline.csv` (quanti secondi di guida baseline).

In breve: il modello guarda **distrazione**, **tempo di recupero**, **stato emotivo/arousal** e **rischio personale di base**.

5 Come lavora, passo per passo

1. **Legge i dati.**
2. **Controlla i dati.** Se trova problemi gravi, si ferma.
3. **Capisce il rischio "normale".** Ogni persona puo avere un rischio base diverso.
4. **Costruisce esempi.**
   - esempi positivi = secondi in cui c'e stato un errore;
   - esempi negativi = secondi in cui non c'e stato errore.
5. **Addestra il modello** (XGBoost), cioe un algoritmo che impara dai dati.

6. **Lo testa su persone non viste** per vedere se davvero generalizza.
7. **Sistema le probabilita** (calibrazione), cosi i numeri sono piu affidabili.
8. **Salva il modello** in un file `.pkl` per usarlo dopo.

6 Come calcola la probabilita di errore

Quando arriva un nuovo istante di guida, il modello guarda:

- se la distrazione e' attiva;
- da quanto tempo e' finita l'ultima distrazione;
- segnali di arousal/emozione;
- rischio base personale del guidatore.

Poi fa due passaggi:

1. produce una probabilita "grezza" (`raw_prob`);
2. la corregge con una calibrazione (`calibrated_prob`).

La probabilita finale di errore e' quella calibrata.

7 Perche fa anche la calibrazione

Perche un modello puo dire "0.70" ma nella realta quel numero potrebbe non essere preciso.
La calibrazione serve a rendere i numeri piu onesti:

- se dice 0.30, dovrebbe davvero significare "circa 30%" di rischio;
- se dice 0.80, dovrebbe davvero significare "rischio molto alto".

8 Metriche spiegate come a un bambino

Le metriche sono voti del modello.

- **Precision**: quando il modello dice "attenzione!", quante volte ha ragione.
- **Recall**: quanti errori veri riesce a trovare.
- **F1**: un voto che mette insieme precision e recall.
- **AUC-PR**: quanto e' bravo a trovare errori rari.
- **AUC-ROC**: quanto separa bene momenti sicuri e rischiosi.
- **Brier**: quanto sono giuste le probabilita (piu basso e meglio).
- **Log-Loss**: punisce tanto le probabilita molto sbagliate.
- **MCC/Kappa**: voti robusti anche quando gli errori sono pochi.
- **Specificity**: quanto e' bravo a riconoscere i secondi sicuri.
- **NPV**: se dice "sei sicuro", quanto spesso e' vero.

In piu, il codice usa il **bootstrap**: rifa i conti tante volte per capire quanto i voti sono stabili.

9 Cosa restituisce alla fine

La funzione `predict_fitness(...)` restituisce:

- `error_probability`: rischio di errore nel prossimo secondo;
- `fitness_to_drive`: quanto sei "in forma" per guidare;
- `alert`: vero/falso, se il rischio supera una soglia;
- `input_warnings`: avvisi se qualche dato in ingresso non va bene.

10 Frase finale

Questo sistema non dice solo "errore/non errore". Dice anche **quanto rischio c'e**, e prova a dirlo in modo affidabile. Per questo usa controlli sui dati, test seri, calibrazione e tante metriche.