Simple Explanation of Fitness-to-Drive

Easy-to-read version

February 22, 2026

1 Super simple idea

Imagine a digital co-driver. This co-driver watches what happens while you drive and tries to answer one question:

"In the next second, how likely is a driving error?"

The file `ftd_predictor.py` builds this co-driver.

2 What Fitness-to-Drive means

Fitness-to-Drive means: *how ready you are to drive safely right now.*

In the code:

```
fitness_to_drive = 1 - error_probability
```

Example:

- if `error_probability = 0.20`, then `fitness_to_drive = 0.80`.
- if `error_probability = 0.70`, then `fitness_to_drive = 0.30`.

So:

- low error probability = safer driving condition;
- high error probability = riskier driving condition.

3 What data it uses

The program reads 4 CSV files:

1. when distractions happened;
2. when errors happened in distraction runs;
3. errors in normal driving (baseline);
4. how many baseline driving seconds we have.

In simple words: it looks at **when you are distracted**, **when you make errors**, and **what happens when there are no special distractions**.

4 The 9 model features (simple explanation)

The model uses 9 clues (features). Each clue is a number that helps estimate risk.

1. `distraction_active`
   Says if a distraction is active now: 1 = yes, 0 = no.
   Computed from `timestamp_start` and `timestamp_end` in `Dataset Distractions_distraction.csv`.

2. `time_since_last_dist`
   Says how many seconds passed since the last distraction ended.
   Also computed from distraction window timestamps.

3. `model_prob`
   Confidence from the arousal model.
   For error samples it comes from `model_prob` in `Dataset Errors_distraction.csv`.
   For distraction negatives it comes from `model_prob_start` or `model_prob_end`.

4. `model_pred_enc`
   Arousal label transformed into a number (encoding).
   Built from `model_pred`, `model_pred_start`, `model_pred_end`.

5. `emotion_prob`
   Confidence of the emotion classifier.
   Built from `emotion_prob` (errors) or `emotion_prob_start/end` (distractions).

6. `emotion_label_enc`
   Emotion label transformed into a number (encoding).
   Built from `emotion_label`, `emotion_label_start`, `emotion_label_end`.

7. `baseline_error_rate`
   The driver's normal risk when driving without special distractions.
   Computed from:

   - `Dataset Errors_baseline.csv` (how many baseline errors);
   - `Dataset Driving Time_baseline.csv` (how many baseline driving seconds).

8. `speed_kmh`
   Car speed at that moment.
   From `speed_kmh` (error rows) or `speed_kmh_start/end` (distraction rows).

9. `steer_angle_deg`
   Steering wheel angle in degrees.
   From `steer_angle_deg` (error rows) or `steer_angle_deg_start/end` (distraction rows).

In short: the model uses **distraction status**, **recovery time**, **emotion/arousal state**, **personal baseline risk**, and **vehicle motion** (speed and steering).

5 How it works, step by step

1. **Read data.**

2. **Check data quality.** If data has serious issues, stop.

3. **Estimate normal risk.** Each driver may have a different baseline risk.

4. **Build training examples.**

   - positive examples = seconds where an error happened;

- negative examples = seconds where no error happened.

5. **Train the model** (XGBoost).
6. **Test on unseen drivers** to check real generalization.
7. **Calibrate probabilities** to make numbers more reliable.
8. **Save the model** in a `.pkl` file.

6 How error probability is computed

For a new driving instant, the model looks at:

- is distraction active or not;
- how long since the last distraction ended;
- arousal/emotion signals;
- speed and steering angle;
- the driver's baseline risk.

Then it does two steps:

1. compute a raw probability (`raw_prob`);
2. adjust it with calibration (`calibrated_prob`).

The final error probability is the calibrated one.

7 Why calibration is used

A model can output "0.70", but that number may not be perfectly honest.
Calibration makes probabilities more trustworthy:

- if it says 0.30, it should behave close to 30% risk;
- if it says 0.80, it should really mean very high risk.

8 Metrics explained for a child

Metrics are model report-card grades.

- **Precision**: when the model says "warning!", how often it is correct.
- **Recall**: how many real errors it catches.
- **F1**: one score that combines precision and recall.
- **AUC-PR**: how good it is at finding rare errors.
- **AUC-ROC**: how well it separates safe vs risky moments.
- **Brier**: how good probability numbers are (lower is better).
- **Log-Loss**: strong penalty for very wrong confident probabilities.
- **MCC/Kappa**: robust scores when classes are imbalanced.
- **Specificity**: how well it recognizes safe seconds.
- **NPV**: when it says "safe", how often it is truly safe.

The code also uses **bootstrap**: it repeats metric computation many times to check stability.

9 What the final function returns

The function `predict_fitness(...)` returns:

- `error_probability`: risk of error in the next second;
- `fitness_to_drive`: how fit/safe the driver is right now;
- `alert`: true/false if risk is above threshold;
- `input_warnings`: warnings for invalid or unusual inputs.

10 Final sentence

This system does not only say "error / no error". It also estimates **how much risk there is**, and tries to make that number reliable using data checks, validation, calibration, and multiple metrics.