

Package: gp3ml (via r-universe)

September 20, 2026

Type Package

Title Governance-First Predictive Modelling for 'Gazepoint' Research

Version 0.3.1

Description Provides governance-first infrastructure for leakage-resistant predictive modelling and validation using 'Gazepoint'-derived research data. Supports explicit task and role declarations, feature-provenance manifests, group-aware holdout splitting and repeated resampling, repository-aware fold evaluation, explicit governed tuning, nested grouped resampling, fold-local preprocessing, discrimination and calibration metrics, target-aligned uncertainty, external-validation and transportability reports, prediction-to-decision governance, target-aware conformal prediction, dataset-shift auditing, locked analysis plans, portable model artifacts, robustness diagnostics, environment provenance, research-object export, model cards, and reproducibility evidence. Intended only for explicitly observed, non-sensitive outcomes and declared scientific purposes. Use is prohibited for person identification, biometric authentication, health or protected-attribute inference, and direct or indirect inference of emotion, stress, personality, deception, cognition, comprehension, intent, or other mental states.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

Depends R (>= 4.1.0)

Suggests bundle, callr, openssl, renv, jsonlite, keras3, knitr, nnet, pkgdown, ranger, rmarkdown, testthat (>= 3.0.0), xgboost

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

URL <https://stefanosbalaskas.github.io/gp3ml/>,
<https://CRAN.R-project.org/package=gp3ml>,
<https://github.com/stefanosbalaskas/gp3ml>

BugReports <https://github.com/stefanosbalaskas/gp3ml/issues>

Repository <https://stefanosbalaskas.r-universe.dev>

Date/Publication 2026-09-20 00:13:37 UTC

RemoteUrl <https://github.com/stefanosbalaskas/gp3ml>

RemoteRef HEAD

RemoteSha 83a8ba6cbc1620c3842087090128cd8971c9d08a

Contents

apply_gazepoint_calibrator	5
apply_gazepoint_decision_rule	6
as_gp3ml_data	7
assert_gp3ml_engine_available	7
assert_gp3ml_use_case	8
assess_gazepoint_calibration	9
assess_gazepoint_conformal_coverage	10
audit_gazepoint_abstention	11
audit_gazepoint_dataset_shift	11
audit_gazepoint_group_folds	12
audit_gazepoint_missingness_shift	13
audit_gazepoint_ml_leakage	14
audit_gazepoint_model_robustness	16
audit_gazepoint_nested_resampling	17
audit_gazepoint_plan_deviations	17
audit_gazepoint_reproducibility	18
audit_gp3ml_api_stability	19
audit_gp3ml_governance_profile	19
bake_gazepoint_preprocessor	20
bootstrap_gazepoint_metrics	21
bootstrap_gazepoint_metrics_by_unit	22
capture_gazepoint_environment	24
collect_gazepoint_fold_predictions	25
combine_gazepoint_handoffs	25
compare_gazepoint_environments	26
compare_gazepoint_models	26
create_external_validation_report	27
create_gazepoint_decision_rule	28
create_gazepoint_feature_manifest	30
create_gazepoint_group_folds	31
create_gazepoint_handoff	34

create_gazepoint_model_artifact	35
create_gazepoint_model_card	36
create_gazepoint_nested_folds	37
create_gazepoint_release_evidence	39
create_gazepoint_release_model_card	40
create_gazepoint_reproducibility_report	41
create_gazepoint_synthetic_manifest	42
create_gazepoint_synthetic_task	43
create_gazepoint_tuning_grid	44
create_gp3ml_governance_profile	45
declare_gazepoint_analysis_plan	45
declare_gazepoint_external_dataset	47
declare_gazepoint_task	48
diagnose_gazepoint_group_folds	50
evaluate_external_validation	52
evaluate_gazepoint_external_transportability	54
evaluate_gazepoint_feature_stability	55
evaluate_gazepoint_group_folds	55
evaluate_gazepoint_missingness_sensitivity	57
evaluate_gazepoint_nested_resampling	58
evaluate_gazepoint_seed_stability	59
evaluate_gazepoint_threshold_stability	59
evaluate_gazepoint_thresholds	60
fit_gazepoint_calibrator	61
fit_gazepoint_conformal	62
fit_gazepoint_deep_model	63
fit_gazepoint_model	65
fit_gazepoint_preprocessor	66
gazepoint_classification_metrics	68
gazepoint_performance_metrics	69
gazepoint_regression_metrics	70
gp3ml_api_contracts	71
gp3ml_available_engines	71
gp3ml_engine_capabilities	72
gp3ml_interop_contracts	72
gp3ml_object_schema	73
gp3ml_prohibited_uses	73
integrate_black_box_model	74
lock_gazepoint_analysis_plan	75
normalize_gazepoint_artifact_text	76
plot_gp3ml_abstention_audit	76
plot_gp3ml_api_stability_audit	77
plot_gp3ml_conformal_coverage	77
plot_gp3ml_dataset_shift_audit	78
plot_gp3ml_engine_capabilities	78
plot_gp3ml_environment_comparison	79
plot_gp3ml_governance_profile_audit	79
plot_gp3ml_handoff_validation	80

plot.gp3ml_model_artifact_validation	80
plot.gp3ml_model_robustness_audit	81
plot.gp3ml_plan_deviation_audit	81
plot.gp3ml_release_checksum_validation	82
plot.gp3ml_reproducibility_audit	82
plot.gp3ml_research_bundle_validation	83
plot.gp3ml_ro_crate_validation	83
plot.gp3ml_threshold_evaluation	84
predict.gp3ml_model	84
predict_gazepoint_interval	86
predict_gazepoint_set	86
print.gazepoint_feature_manifest_validation	87
print.gazepoint_fold_diagnostics	87
print.gazepoint_fold_diagnostics_validation	89
print.gazepoint_group_folds	91
print.gazepoint_group_folds_audit	93
print.gazepoint_group_folds_validation	94
print.gazepoint_ml_leakage_audit	96
print.gazepoint_ml_split	97
print.gazepoint_ml_split_validation	99
restore_gazepoint_model_artifact	101
select_gazepoint_model	101
select_gazepoint_threshold	102
simulate_gazepoint_governed_data	103
simulate_gazepoint_research_handoffs	104
split_gazepoint_ml_data	105
summarize_gazepoint_resample_performance	107
summarize_gazepoint_resample_uncertainty	108
summarize_gazepoint_shift	109
test_gazepoint_model_portability	109
train_gazepoint_classifier	110
tune_gazepoint_model	111
validate_gazepoint_analysis_plan	112
validate_gazepoint_conformal	113
validate_gazepoint_decision_rule	113
validate_gazepoint_environment	114
validate_gazepoint_feature_manifest	114
validate_gazepoint_fold_diagnostics	115
validate_gazepoint_group_folds	117
validate_gazepoint_handoff	118
validate_gazepoint_ml_roles	119
validate_gazepoint_ml_split	121
validate_gazepoint_model_artifact	122
validate_gazepoint_model_tuning	123
validate_gazepoint_nested_evaluation	123
validate_gazepoint_nested_folds	124
validate_gazepoint_release_checksums	124
validate_gazepoint_resample_evaluation	125

validate_gazepoint_research_bundle	125
validate_gazepoint_ro_crate	126
validate_gazepoint_target_uncertainty	126
validate_gazepoint_transportability	127
validate_gp3ml_object_contract	127
with_gazepoint_reproducible_output	128
write_external_validation_report	128
write_gazepoint_analysis_plan	130
write_gazepoint_feature_manifest_csv	130
write_gazepoint_fold_diagnostics_csv	131
write_gazepoint_group_folds_csv	133
write_gazepoint_ml_leakage_audit_csv	136
write_gazepoint_ml_split_csv	137
write_gazepoint_model_card	139
write_gazepoint_model_tuning	141
write_gazepoint_nested_evaluation	142
write_gazepoint_release_checksums	142
write_gazepoint_release_model_card	143
write_gazepoint_reproducibility_audit	143
write_gazepoint_reproducibility_report	144
write_gazepoint_resample_evaluation	145
write_gazepoint_ro_crate	145
write_gazepoint_target_uncertainty	146
write_gazepoint_transportability_report	147
write_gp3ml_api_contracts	148
write_gp3ml_governance_profile	148

Index 150

apply_gazepoint_calibrator

Apply a fitted probability calibrator

Description

Apply a fitted probability calibrator

Usage

```
apply_gazepoint_calibrator(calibrator, probability)
```

Arguments

calibrator	A fitted gp3ml_calibrator object.
probability	Uncalibrated probabilities to transform.

Value

A numeric vector of calibrated probabilities, clipped to the open unit interval.

Examples

```
truth <- factor(
  rep(c("pass", "review"), 6),
  levels = c("pass", "review")
)
probability <- c(
  0.20, 0.70, 0.60, 0.55, 0.30, 0.80,
  0.65, 0.45, 0.40, 0.75, 0.50, 0.60
)
calibrator <- fit_gazepoint_calibrator(
  truth = truth,
  probability = probability,
  positive = "review"
)
apply_gazepoint_calibrator(
  calibrator,
  probability
)
```

apply_gazepoint_decision_rule

Apply a governed classification decision rule

Description

Apply a governed classification decision rule

Usage

```
apply_gazepoint_decision_rule(
  rule,
  probability,
  positive,
  negative,
  abstain_label = ".abstain"
)
```

Arguments

rule	A validated decision rule.
probability	Positive-class probability.
positive	Positive class label.
negative	Negative class label.
abstain_label	Label used for abstentions.

Value

A factor of governed decisions.

as_gp3ml_data	<i>Extract model-ready data from a gp3ml handoff object</i>
---------------	---

Description

Extract model-ready data from a gp3ml handoff object

Usage

```
as_gp3ml_data(x, ...)
```

Arguments

x	A gp3ml_handoff or gp3ml_handoff_bundle.
...	Reserved for future adapters.

Value

A data frame.

assert_gp3ml_engine_available	<i>Assert that a gp3ml engine is available</i>
-------------------------------	--

Description

Assert that a gp3ml engine is available

Usage

```
assert_gp3ml_engine_available(engine, check_keras_backend = FALSE)
```

Arguments

engine	Engine name.
check_keras_backend	Whether to verify a configured Keras backend.

Value

Invisibly TRUE on success.

assert_gp3ml_use_case *Assert that a task is within the permitted gp3ml scope*

Description

Assert that a task is within the permitted gp3ml scope

Usage

```
assert_gp3ml_use_case(task, data = NULL)
```

Arguments

task	A gp3ml_task object.
data	Optional data frame used to validate task columns.

Value

Invisibly returns TRUE when the task is permitted; otherwise, the function stops with an error.

Examples

```
example_data <- data.frame(
  participant_id = rep(sprintf("P%02d", 1:12), each = 2),
  trial_id = sprintf("T%02d", 1:24),
  stimulus_id = rep(c("S01", "S02"), 12),
  condition = rep(c("A", "B"), 12),
  fixation_duration = 180 + seq_len(24),
  pupil_change = sin(seq_len(24) / 3),
  stringsAsFactors = FALSE
)
example_data$quality_status <- factor(
  c(
    "pass", "review", "pass", "review", "review", "pass",
    "review", "pass", "pass", "review", "review", "pass",
    "review", "pass", "review", "pass", "pass", "review",
    "pass", "review", "review", "pass", "pass", "review"
  ),
  levels = c("pass", "review")
)
task <- declare_gazepoint_task(
  data = example_data,
  outcome = "quality_status",
  purpose = "Predict predefined recording-quality review status",
  task_type = "classification",
  unit_id = "trial_id",
  participant_id = "participant_id",
  stimulus_id = "stimulus_id",
  generalization_target = "new_participants",
```

```
    positive = "review"
  )
  assert_gp3ml_use_case(task, example_data)
```

assess_gazepoint_calibration

Calibration assessment with bootstrap uncertainty

Description

Calibration assessment with bootstrap uncertainty

Usage

```
assess_gazepoint_calibration(
  truth,
  probability,
  positive = NULL,
  bins = 10L,
  bootstrap = 200L,
  conf_level = 0.95,
  seed = 1L
)
```

Arguments

truth	Observed binary outcome values.
probability	Predicted positive-class probabilities.
positive	Label representing the positive class.
bins	Number of reliability bins.
bootstrap	Number of bootstrap replicates.
conf_level	Confidence level for percentile intervals.
seed	Deterministic random seed.

Value

A `gp3ml_calibration_assessment` object containing calibration summaries, reliability-bin results, bootstrap intervals, and assessment settings.

Examples

```
truth <- factor(
  rep(rep(c("pass", "review"), 5), 10),
  levels = c("pass", "review")
)
probability <- rep(
```

```

    seq(0.10, 0.90, length.out = 10),
    each = 10
  )

  assessment <- assess_gazepoint_calibration(
    truth = truth,
    probability = probability,
    positive = "review",
    bins = 5L,
    bootstrap = 10L,
    seed = 101L
  )
  assessment

```

```

assess_gazepoint_conformal_coverage
Assess conformal coverage

```

Description

Assess conformal coverage

Usage

```

assess_gazepoint_conformal_coverage(
  object,
  truth,
  interval = NULL,
  set = NULL,
  unit = NULL
)

```

Arguments

<code>object</code>	A <code>gp3ml_conformal_fit</code> .
<code>truth</code>	Observed outcomes.
<code>interval</code>	Regression interval data frame.
<code>set</code>	Classification set data frame.
<code>unit</code>	Optional assessment-unit identifier.

Value

A `gp3ml_conformal_coverage`.

```
audit_gazepoint_abstention
```

Audit abstention decisions

Description

Audit abstention decisions

Usage

```
audit_gazepoint_abstention(truth, decision, abstain_label = ".abstain")
```

Arguments

truth	Observed binary outcome.
decision	Decisions returned by <code>apply_gazepoint_decision_rule()</code> .
abstain_label	Abstention label.

Value

A `gp3ml_abstention_audit`.

```
audit_gazepoint_dataset_shift
```

Audit predictor distribution shift

Description

Audit predictor distribution shift

Usage

```
audit_gazepoint_dataset_shift(
  development,
  external,
  predictors = intersect(names(development), names(external)),
  thresholds = list(smd_review = 0.2, smd_fail = 0.5, outside_review = 0.05, outside_fail
    = 0.2, tv_review = 0.2, tv_fail = 0.4)
)
```

Arguments

development	Development/training data.
external	Independent or later data to compare.
predictors	Predictors to audit. Defaults to common columns.
thresholds	Named threshold list.

Value

A gp3ml_dataset_shift_audit.

audit_gazepoint_group_folds

Aggregate leakage audits across group-aware folds

Description

Aggregate leakage audits across group-aware folds

Usage

```
audit_gazepoint_group_folds(x)
```

Arguments

x A gazepoint_group_folds object.

Value

An object of class gazepoint_group_folds_audit.

Examples

```
example_data <- expand.grid(
  participant_id = sprintf("P%02d", 1:6),
  stimulus_id = sprintf("S%02d", 1:4),
  repetition = 1:2,
  KEEP.OUT.ATTRS = FALSE,
  stringsAsFactors = FALSE
)
example_data$trial_id <- paste0(
  example_data$stimulus_id,
  "_T",
  example_data$repetition
)
participant_number <- as.integer(
  sub("P", "", example_data$participant_id)
)
stimulus_number <- as.integer(
  sub("S", "", example_data$stimulus_id)
)
example_data$outcome <- factor(
  ifelse(
    (participant_number + stimulus_number) %% 2L == 0L,
    "review",
    "pass"
  ),
```

```

    levels = c("pass", "review")
  )
  row_index <- seq_len(nrow(example_data))
  example_data$fixation_duration <- 180 + row_index
  example_data$pupil_change <- round(
    sin(row_index / 7),
    4
  )
  )
  example_data$repetition <- NULL
  manifest <- create_gazepoint_feature_manifest(
    features = c("fixation_duration", "pupil_change"),
    scientific_source = c(
      "Gazepoint fixation export",
      "Gazepoint pupil export"
    ),
  ),
  source_table = c("fixations", "pupil"),
  transformation = c(
    "Trial-level mean",
    "Trial-level change"
  ),
  ),
  availability_stage = "during_exposure",
  prediction_time_available = TRUE,
  preprocessing_scope = "none",
  fold_local_required = FALSE
)
folds <- create_gazepoint_group_folds(
  data = example_data,
  outcome = "outcome",
  predictors = c("fixation_duration", "pupil_change"),
  feature_manifest = manifest,
  generalization_target = "new_participants",
  participant_id = "participant_id",
  trial_id = "trial_id",
  stimulus_id = "stimulus_id",
  v = 3L,
  repeats = 1L,
  seed = 101L
)
audit <- audit_gazepoint_group_folds(folds)
audit

```

audit_gazepoint_missingness_shift

Audit missingness shift

Description

Audit missingness shift

Usage

```
audit_gazepoint_missingness_shift(
  development,
  external,
  predictors = intersect(names(development), names(external)),
  review_delta = 0.1,
  fail_delta = 0.25
)
```

Arguments

development	Development data.
external	External/new data.
predictors	Predictors to audit.
review_delta	Review threshold for absolute missingness change.
fail_delta	Fail threshold for absolute missingness change.

Value

A gp3ml_missingness_shift_audit.

audit_gazepoint_ml_leakage

Audit leakage between predictive-analysis partitions

Description

Audits already-defined analysis and assessment partitions for common forms of leakage and for incompatibility with a declared generalization target. The function does not create data splits, pre-process variables, select features, or fit predictive models.

Usage

```
audit_gazepoint_ml_leakage(
  analysis,
  assessment,
  outcome,
  predictors,
  participant_id = NULL,
  trial_id = NULL,
  stimulus_id = NULL,
  generalization_target = c("new_trials_known_participants", "new_participants",
    "new_stimuli", "new_participants_and_new_stimuli"),
  target_derived = character(),
  post_outcome = character()
)
```

Arguments

analysis	A data frame containing the analysis or training partition.
assessment	A data frame containing the assessment or test partition.
outcome	A single column name identifying the outcome.
predictors	A character vector identifying intended predictor columns.
participant_id	An optional participant-identifier column.
trial_id	An optional trial-identifier column.
stimulus_id	An optional stimulus-identifier column.
generalization_target	The predictive generalization target. One of "new_trials_known_participants", "new_participants", "new_stimuli", or "new_participants_and_new_stimuli".
target_derived	Character vector of columns known to have been derived directly from the outcome.
post_outcome	Character vector of columns measured or constructed after the outcome became available.

Details

The overall status is "fail" when at least one failing check is present, "review" when no failing checks are present but at least one review item is present, and "pass" otherwise.

When participant_id is supplied, trial overlap is evaluated using composite participant-trial units. This permits trial labels such as "T01" to be reused by different participants without being treated as leakage. Without participant_id, trial_id is assumed to be globally unique.

The audit can identify structural leakage visible in the supplied partitions and declared variable roles. It cannot prove that preprocessing or feature selection was estimated inside resampling folds. Those operations require separate provenance and resampling safeguards.

The function does not determine whether an outcome is scientifically or ethically appropriate. All uses remain subject to the package governance and prohibited-use statements.

Value

An object of class gazepoint_ml_leakage_audit. The object contains an overall status, partition summary, complete check table, and machine-readable table of non-passing issues.

Examples

```
analysis <- data.frame(
  participant_id = c("P01", "P01", "P02", "P02"),
  trial_id = c("T01", "T02", "T03", "T04"),
  stimulus_id = c("S01", "S02", "S03", "S04"),
  outcome = c(0, 1, 0, 1),
  fixation_duration = c(210, 240, 225, 260),
  pupil_change = c(0.10, 0.16, 0.12, 0.18)
)

assessment <- data.frame(
```

```

    participant_id = c("P03", "P03", "P04", "P04"),
    trial_id = c("T05", "T06", "T07", "T08"),
    stimulus_id = c("S05", "S06", "S07", "S08"),
    outcome = c(1, 0, 1, 0),
    fixation_duration = c(275, 230, 290, 245),
    pupil_change = c(0.21, 0.11, 0.24, 0.14)
)

audit_gazepoint_ml_leakage(
  analysis = analysis,
  assessment = assessment,
  outcome = "outcome",
  predictors = c("fixation_duration", "pupil_change"),
  participant_id = "participant_id",
  trial_id = "trial_id",
  stimulus_id = "stimulus_id",
  generalization_target = "new_participants"
)

```

audit_gazepoint_model_robustness

Audit multiple robustness dimensions

Description

Audit multiple robustness dimensions

Usage

```

audit_gazepoint_model_robustness(
  seed_stability = NULL,
  feature_stability = NULL,
  threshold_stability = NULL,
  missingness_stability = NULL,
  relative_sd_review = 0.05,
  relative_sd_fail = 0.15
)

```

Arguments

`seed_stability` Optional seed-stability object.

`feature_stability`
Optional feature-stability object.

`threshold_stability`
Optional threshold-stability object.

`missingness_stability`
Optional missingness-stability object.

relative_sd_review
 Relative SD threshold for review.

relative_sd_fail
 Relative SD threshold for fail.

Value

A gp3ml_model_robustness_audit.

audit_gazepoint_nested_resampling
 Audit nested grouped resampling for outer-assessment leakage

Description

Audit nested grouped resampling for outer-assessment leakage

Usage

```
audit_gazepoint_nested_resampling(x)

## S3 method for class 'gp3ml_nested_resampling_audit'
print(x, ...)
```

Arguments

x A gp3ml_nested_folds object.

... Additional arguments passed to the print method.

Value

A gp3ml_nested_resampling_audit.

audit_gazepoint_plan_deviations
 Audit deviations from a locked analysis plan

Description

Audit deviations from a locked analysis plan

Usage

```
audit_gazepoint_plan_deviations(
  plan,
  actual,
  fields = c("outcome", "predictors", "generalization_target", "primary_metric",
    "secondary_metrics", "calibration_metric", "uncertainty_method", "threshold_policy",
    "candidate_models", "preprocessing_plan")
)
```

Arguments

plan	A locked analysis plan.
actual	Named list describing the analysis actually performed.
fields	Fields to compare.

Value

A gp3ml_plan_deviation_audit.

audit_gazepoint_reproducibility

Audit generated artifacts for volatile output

Description

Audit generated artifacts for volatile output

Usage

```
audit_gazepoint_reproducibility(
  paths,
  recursive = TRUE,
  extensions = c("R", "Rmd", "Rd", "md", "txt", "html", "json", "csv", "yaml", "yml")
)
```

Arguments

paths	Files or directories to audit.
recursive	Whether directories are searched recursively.
extensions	Text-file extensions to inspect.

Value

A gp3ml_reproducibility_audit.

`audit_gp3ml_api_stability`*Audit gp3ml API stability*

Description

Audit gp3ml API stability

Usage

```
audit_gp3ml_api_stability(registry = gp3ml_api_contracts())
```

Arguments

registry Contract registry.

Value

A gp3ml_api_stability_audit object.

`audit_gp3ml_governance_profile`*Audit a governance-evidence profile*

Description

Audit a governance-evidence profile

Usage

```
audit_gp3ml_governance_profile(profile)
```

Arguments

profile Governance profile.

Value

A gp3ml_governance_profile_audit.

bake_gazepoint_preprocessor

Apply a fitted preprocessing engine

Description

Apply a fitted preprocessing engine

Usage

```
bake_gazepoint_preprocessor(preprocessor, new_data)
```

Arguments

preprocessor A fitted gp3ml_preprocessor object.
new_data Data to transform using the fitted parameters.

Value

A numeric model matrix transformed using only the parameters stored in the fitted preprocessor.

Examples

```
example_data <- data.frame(
  participant_id = rep(sprintf("P%02d", 1:12), each = 2),
  trial_id = sprintf("T%02d", 1:24),
  stimulus_id = rep(c("S01", "S02"), 12),
  condition = rep(c("A", "B"), 12),
  fixation_duration = 180 + seq_len(24),
  pupil_change = sin(seq_len(24) / 3),
  stringsAsFactors = FALSE
)
example_data$quality_status <- factor(
  c(
    "pass", "review", "pass", "review", "review", "pass",
    "review", "pass", "pass", "review", "review", "pass",
    "review", "pass", "review", "pass", "pass", "review",
    "pass", "review", "review", "pass", "pass", "review"
  ),
  levels = c("pass", "review")
)
preprocessor <- fit_gazepoint_preprocessor(
  data = example_data,
  predictors = c(
    "fixation_duration",
    "pupil_change",
    "condition"
  )
)
```

```
baked <- bake_gazepoint_preprocessor(
  preprocessor,
  example_data
)
dim(baked)
```

bootstrap_gazepoint_metrics

Bootstrap uncertainty intervals for performance metrics

Description

Bootstrap uncertainty intervals for performance metrics

Usage

```
bootstrap_gazepoint_metrics(
  task,
  truth,
  prediction = NULL,
  probability = NULL,
  threshold = 0.5,
  bootstrap = 1000L,
  conf_level = 0.95,
  seed = 1L
)
```

Arguments

task	A governed gp3ml_task object.
truth	Observed outcome values.
prediction	Predicted classes or numeric values.
probability	Predicted positive-class probabilities.
threshold	Probability threshold for classification.
bootstrap	Number of bootstrap replicates.
conf_level	Confidence level for percentile intervals.
seed	Deterministic random seed.

Value

A gp3ml_metric_uncertainty object containing point estimates, percentile intervals, bootstrap draws, resampling settings, and the governed task.

Examples

```

example_data <- data.frame(
  participant_id = rep(sprintf("P%02d", 1:12), each = 2),
  trial_id = sprintf("T%02d", 1:24),
  stimulus_id = rep(c("S01", "S02"), 12),
  condition = rep(c("A", "B"), 12),
  fixation_duration = 180 + seq_len(24),
  pupil_change = sin(seq_len(24) / 3),
  stringsAsFactors = FALSE
)
example_data$quality_status <- factor(
  c(
    "pass", "review", "pass", "review", "review", "pass",
    "review", "pass", "pass", "review", "review", "pass",
    "review", "pass", "review", "pass", "pass", "review",
    "pass", "review", "review", "pass", "pass", "review"
  ),
  levels = c("pass", "review")
)
task <- declare_gazeport_task(
  data = example_data,
  outcome = "quality_status",
  purpose = "Predict predefined recording-quality review status",
  task_type = "classification",
  unit_id = "trial_id",
  participant_id = "participant_id",
  stimulus_id = "stimulus_id",
  generalization_target = "new_participants",
  positive = "review"
)
probability <- seq(
  0.20,
  0.80,
  length.out = nrow(example_data)
)
predicted <- factor(
  ifelse(probability >= 0.5, "review", "pass"),
  levels = levels(example_data$quality_status)
)
uncertainty <- bootstrap_gazeport_metrics(
  task = task,
  truth = example_data$quality_status,
  prediction = predicted,
  probability = probability,
  bootstrap = 10L,
  seed = 101L
)
uncertainty

```

bootstrap_gazepoint_metrics_by_unit

*Generalization-target-aligned bootstrap uncertainty***Description**

Resamples observations or declared clusters while preserving every row that belongs to a sampled cluster. Repeated cluster draws duplicate all associated rows. The returned object records the resampling unit and must not be described as uncertainty for another unit.

Usage

```
bootstrap_gazepoint_metrics_by_unit(
  task,
  truth,
  prediction = NULL,
  probability = NULL,
  participant_id = NULL,
  stimulus_id = NULL,
  unit = c("observation", "participant", "stimulus", "participant_and_stimulus"),
  bootstrap = 1000L,
  conf_level = 0.95,
  seed = 1L,
  threshold = 0.5,
  stratify_observations = TRUE
)

## S3 method for class 'gp3ml_target_uncertainty'
print(x, ...)
```

Arguments

task	Governed task.
truth	Observed outcomes.
prediction	Predicted classes or numeric outcomes.
probability	Positive-class probabilities.
participant_id	Participant identifiers for participant-based methods.
stimulus_id	Stimulus identifiers for stimulus-based methods.
unit	Resampling unit.
bootstrap	Number of replicates.
conf_level	Percentile interval level.
seed	Deterministic seed.
threshold	Classification threshold.
stratify_observations	Whether the observation-level classification bootstrap preserves class counts.

x	An object returned by the corresponding gp3ml constructor, evaluator, summarizer, or validator.
...	Additional arguments passed to the print method.

Value

A gp3ml_target_uncertainty object.

Examples

```
data <- simulate_gazepoint_governed_data(12L, 4L, 1L, 404L)
task <- create_gazepoint_synthetic_task(data, "recording_quality", "new_participants")
probability <- seq(0.15, 0.85, length.out = nrow(data))
prediction <- factor(
  ifelse(probability >= 0.5, "review", "pass"),
  levels = levels(data$quality_status)
)
uncertainty <- bootstrap_gazepoint_metrics_by_unit(
  task,
  truth = data$quality_status,
  prediction = prediction,
  probability = probability,
  participant_id = data$participant_id,
  unit = "participant",
  bootstrap = 20L,
  seed = 404L
)
uncertainty
```

capture_gazepoint_environment

Capture a reproducibility environment record

Description

Capture a reproducibility environment record

Usage

```
capture_gazepoint_environment(
  packages = unique(c("gp3ml", loadedNamespaces())),
  root = ".",
  include_renv = FALSE
)
```

Arguments

packages	Packages to record. Defaults to gp3ml and currently loaded namespaces.
root	Repository/project root used to capture Git SHA.
include_renv	Whether to record an existing renv.lock hash.

Value

A gp3ml_environment_record.

collect_gazepoint_fold_predictions

Collect predictions from a grouped-fold evaluation

Description

Collect predictions from a grouped-fold evaluation

Usage

```
collect_gazepoint_fold_predictions(x, include_failed = TRUE)
```

Arguments

x A gp3ml_resample_evaluation.

include_failed Whether failed folds are represented by explicit status rows when they produced no predictions.

Value

A data frame of row-level assessment predictions with fold labels.

combine_gazepoint_handoffs

Combine validated cross-package handoffs

Description

Combine validated cross-package handoffs

Usage

```
combine_gazepoint_handoffs(
  handoffs,
  keys = NULL,
  collision = c("error", "prefix")
)
```

Arguments

handoffs Named list of gp3ml_handoff objects.

keys Optional join keys; defaults to the first handoff's keys.

collision How to handle overlapping non-key column names.

Value

A gp3ml_handoff_bundle.

compare_gazepoint_environments

Compare two environment records

Description

Compare two environment records

Usage

```
compare_gazepoint_environments(reference, current)
```

Arguments

reference	Reference environment.
current	Current environment.

Value

A gp3ml_environment_comparison.

compare_gazepoint_models

Compare governed model candidates without selecting a winner

Description

Compare governed model candidates without selecting a winner

Usage

```
compare_gazepoint_models(x, metrics = NULL)
```

Arguments

x	A gp3ml_model_tuning object.
metrics	Optional metric names.

Value

A data frame retaining candidate status, failures, complexity, interpretability, and fold-distribution summaries.

```
create_external_validation_report
```

Create an external-validation report object

Description

Create an external-validation report object

Usage

```
create_external_validation_report(
  validation,
  development_metrics = NULL,
  limitations = character()
)
```

Arguments

`validation` A `gp3ml_external_validation` object.

`development_metrics` Optional development-sample metrics.

`limitations` Character vector describing report limitations.

Value

A `gp3ml_external_validation_report` object containing the validation result, optional development metrics, limitations, and prohibited-use information.

Examples

```
training_data <- data.frame(
  participant_id = rep(sprintf("P%02d", 1:12), each = 2),
  trial_id = sprintf("T%02d", 1:24),
  stimulus_id = rep(c("S01", "S02"), 12),
  fixation_duration = 180 + seq_len(24),
  pupil_change = sin(seq_len(24) / 3),
  stringsAsFactors = FALSE
)
training_data$quality_status <- factor(
  c(
    "pass", "review", "pass", "review", "review", "pass",
    "review", "pass", "pass", "review", "review", "pass",
    "review", "pass", "review", "pass", "pass", "review",
    "pass", "review", "review", "pass", "pass", "review"
  ),
  levels = c("pass", "review")
)
task <- declare_gazepoint_task(
```

```

    data = training_data,
    outcome = "quality_status",
    purpose = "Predict predefined recording-quality review status",
    task_type = "classification",
    unit_id = "trial_id",
    participant_id = "participant_id",
    stimulus_id = "stimulus_id",
    generalization_target = "new_participants",
    positive = "review"
  )
  model <- train_gazepoint_classifier(
    data = training_data,
    task = task,
    predictors = c("fixation_duration", "pupil_change"),
    engine = "glm",
    seed = 101L
  )
  external_data <- training_data
  external_data$participant_id <- rep(
    sprintf("E%02d", 1:12),
    each = 2
  )
  external_data$trial_id <- sprintf("ET%02d", 1:24)
  external_data$fixation_duration <-
    external_data$fixation_duration + 4
  external_data$pupil_change <- cos(seq_len(24) / 4)
  validation <- evaluate_external_validation(
    model = model,
    external_data = external_data,
    label = "synthetic_external",
    bootstrap = 10L,
    seed = 101L
  )
  report <- create_external_validation_report(
    validation = validation,
    limitations = "Synthetic external-validation example."
  )
  report

```

```
create_gazepoint_decision_rule
```

Create a governed classification decision rule

Description

Create a governed classification decision rule

Usage

```
create_gazepoint_decision_rule(
```

```

    metric,
    direction = c("maximize", "minimize"),
    threshold = NULL,
    threshold_origin = c("predeclared", "training", "inner_resampling"),
    cost_false_positive = 1,
    cost_false_negative = 1,
    abstention_allowed = FALSE,
    abstention_interval = NULL,
    calibration_source = "none",
    training_partition = "analysis",
    generalization_target,
    scientific_justification
  )

```

Arguments

<code>metric</code>	Metric used to justify the threshold.
<code>direction</code>	Either "maximize" or "minimize".
<code>threshold</code>	Optional probability threshold. Leave NULL until selected.
<code>threshold_origin</code>	Origin of the threshold.
<code>cost_false_positive</code>	Non-negative false-positive cost.
<code>cost_false_negative</code>	Non-negative false-negative cost.
<code>abstention_allowed</code>	Whether abstention is permitted.
<code>abstention_interval</code>	Optional length-two probability interval. Probabilities inside the interval are labelled as abstentions.
<code>calibration_source</code>	Description of the calibration source.
<code>training_partition</code>	Description of the data partition used to determine the threshold.
<code>generalization_target</code>	Declared generalization target.
<code>scientific_justification</code>	Explicit scientific justification.

Value

A `gp3ml_decision_rule`.

```
create_gazepoint_feature_manifest
```

Create a Gazepoint feature-provenance manifest

Description

Creates a structured provenance manifest for intended predictive features. Each row records where a feature originated, when it became available, whether it is outcome-derived or post-outcome, and where any data-dependent preprocessing was estimated.

Usage

```
create_gazepoint_feature_manifest(
  features,
  scientific_source = NA_character_,
  source_table = NA_character_,
  transformation = "none",
  availability_stage = "unknown",
  prediction_time_available = NA,
  outcome_derived = FALSE,
  post_outcome = FALSE,
  identifier = FALSE,
  preprocessing_scope = "unknown",
  fold_local_required = NA,
  reviewer_notes = ""
)
```

Arguments

<code>features</code>	Character vector of unique feature names.
<code>scientific_source</code>	Scientific or measurement source for each feature.
<code>source_table</code>	Source export, table, or object for each feature.
<code>transformation</code>	Description of the transformation used to construct each feature.
<code>availability_stage</code>	Availability stage for each feature. One of "pre_exposure", "during_exposure", "post_exposure_pre_outcome", "at_prediction", "post_outcome", or "unknown".
<code>prediction_time_available</code>	Logical vector indicating whether each feature is available at the intended prediction time.
<code>outcome_derived</code>	Logical vector indicating whether each feature was derived directly or indirectly from the outcome.
<code>post_outcome</code>	Logical vector indicating whether each feature was measured or constructed after the outcome became available.

identifier Logical vector indicating whether each feature is an identifier or row-location variable.

preprocessing_scope Scope in which any data-dependent preprocessing was estimated. One of "none", "global", "analysis_partition", "resampling_fold", or "unknown".

fold_local_required Logical vector indicating whether preprocessing for each feature must be estimated separately inside each resampling fold.

reviewer_notes Optional reviewer-facing notes.

Details

Each row is treated as an intended predictor. Consequently, outcome-derived, post-outcome, unavailable, and identifier features are treated as failing conditions by [validate_gazepoint_feature_manifest\(\)](#).

The manifest records declared provenance. It does not independently prove that preprocessing was estimated within the stated scope.

Value

A data frame of class `gazepoint_feature_manifest`.

Examples

```
manifest <- create_gazepoint_feature_manifest(
  features = c("fixation_duration", "pupil_change"),
  scientific_source = c(
    "Gazepoint fixation export",
    "Gazepoint all-gaze export"
  ),
  source_table = c("fixations", "all_gaze"),
  transformation = c(
    "Trial-level mean",
    "Baseline-adjusted change"
  ),
  availability_stage = "during_exposure",
  prediction_time_available = TRUE,
  preprocessing_scope = c("none", "resampling_fold"),
  fold_local_required = c(FALSE, TRUE)
)

manifest
```

create_gazepoint_group_folds

Create deterministic group-aware Gazepoint resampling folds

Description

Creates repeated grouped assessment folds that preserve the grouping structure implied by an explicit generalization target. A passing feature-provenance manifest is required, and every analysis-assessment pair is evaluated using the leakage audit.

Usage

```
create_gazepoint_group_folds(
  data,
  outcome,
  predictors,
  feature_manifest,
  generalization_target,
  participant_id = NULL,
  trial_id = NULL,
  stimulus_id = NULL,
  v = 5L,
  repeats = 1L,
  seed = 1L,
  source_row_id = ".gp3ml_source_row"
)
```

Arguments

<code>data</code>	A data frame containing the outcome, predictors, and grouping identifiers.
<code>outcome</code>	Name of the outcome column.
<code>predictors</code>	Character vector naming intended predictors.
<code>feature_manifest</code>	A feature manifest containing all intended predictors.
<code>generalization_target</code>	One of "new_trials_known_participants", "new_participants", "new_stimuli", or "new_participants_and_new_stimuli".
<code>participant_id</code>	Optional participant-identifier column.
<code>trial_id</code>	Optional trial-identifier column.
<code>stimulus_id</code>	Optional stimulus-identifier column.
<code>v</code>	Number of group folds. For simultaneous participant and stimulus generalization, a length-two vector specifies participant and stimulus fold counts.
<code>repeats</code>	Number of repeated fold assignments.
<code>seed</code>	Integer random seed. The caller's random-number state is restored.
<code>source_row_id</code>	Name of the source-row identifier added to returned partitions.

Details

For new trials among known participants, participant-trial units are assigned separately within each participant. For simultaneous participant and stimulus generalization, crossed participant-stimulus

assessment blocks are created; cross-block rows are excluded from that fold. Each source row appears in assessment exactly once per repeat.

This function does not perform preprocessing, feature selection, tuning, nested resampling, or model fitting.

Value

An object of class `gazepoint_group_folds`.

Examples

```
example_data <- expand.grid(
  participant_id = sprintf("P%02d", 1:6),
  stimulus_id = sprintf("S%02d", 1:4),
  repetition = 1:2,
  KEEP.OUT.ATTRS = FALSE,
  stringsAsFactors = FALSE
)
example_data$trial_id <- paste0(
  example_data$stimulus_id,
  "_T",
  example_data$repetition
)
participant_number <- as.integer(
  sub("P", "", example_data$participant_id)
)
stimulus_number <- as.integer(
  sub("S", "", example_data$stimulus_id)
)
example_data$outcome <- factor(
  ifelse(
    (participant_number + stimulus_number) %% 2L == 0L,
    "review",
    "pass"
  ),
  levels = c("pass", "review")
)
row_index <- seq_len(nrow(example_data))
example_data$fixation_duration <- 180 + row_index
example_data$pupil_change <- round(
  sin(row_index / 7),
  4
)
example_data$repetition <- NULL
manifest <- create_gazepoint_feature_manifest(
  features = c("fixation_duration", "pupil_change"),
  scientific_source = c(
    "Gazepoint fixation export",
    "Gazepoint pupil export"
  ),
  source_table = c("fixations", "pupil"),
  transformation = c(
```

```

      "Trial-level mean",
      "Trial-level change"
    ),
    availability_stage = "during_exposure",
    prediction_time_available = TRUE,
    preprocessing_scope = "none",
    fold_local_required = FALSE
  )
  folds <- create_gazepoint_group_folds(
    data = example_data,
    outcome = "outcome",
    predictors = c("fixation_duration", "pupil_change"),
    feature_manifest = manifest,
    generalization_target = "new_participants",
    participant_id = "participant_id",
    trial_id = "trial_id",
    stimulus_id = "stimulus_id",
    v = 3L,
    repeats = 1L,
    seed = 101L
  )
  folds

```

```
create_gazepoint_handoff
```

Create a lightweight cross-package Gazepoint handoff

Description

Create a lightweight cross-package Gazepoint handoff

Usage

```

create_gazepoint_handoff(
  data,
  source_package,
  source_version = NULL,
  producer = NULL,
  keys,
  outcome = NULL,
  predictors = character(),
  feature_manifest = NULL,
  notes = character()
)

```

Arguments

data Prepared data frame.

source_package Upstream source package or study_design/custom.

source_version	Optional source-package version.
producer	Optional upstream function/workflow label.
keys	Character vector of row-identifying join keys.
outcome	Optional observed outcome column.
predictors	Optional prepared predictor columns.
feature_manifest	Optional gp3ml feature-provenance manifest.
notes	Optional handoff notes.

Value

A gp3ml_handoff object.

```
create_gazepoint_model_artifact
```

Create a portable governed model artifact

Description

Create a portable governed model artifact

Usage

```
create_gazepoint_model_artifact(
  model,
  preprocessor = model$preprocessor %||% NULL,
  feature_manifest = NULL,
  task = model$task %||% NULL,
  decision_rule = NULL,
  model_card = NULL,
  reference_data = NULL,
  bundle_model = TRUE
)
```

Arguments

model	A fitted gp3ml_model or controlled model object.
preprocessor	Optional preprocessing object.
feature_manifest	Optional feature manifest.
task	Optional task; defaults to model\$task.
decision_rule	Optional decision rule.
model_card	Optional model card.
reference_data	Optional deterministic prediction fixture.
bundle_model	Whether to attempt bundle::bundle() when available.

Value

A gp3ml_model_artifact.

create_gazepoint_model_card
<i>Create a governance-focused model card</i>

Description

Create a governance-focused model card

Usage

```
create_gazepoint_model_card(  
  model,  
  intended_use,  
  evaluation = NULL,  
  calibration = NULL,  
  feature_manifest = NULL,  
  external_validation = NULL,  
  limitations = character(),  
  ethical_review = NULL  
)
```

Arguments

- model A fitted gp3ml_model object.
- intended_use Explicit description of the intended research use.
- evaluation Optional performance-evaluation object.
- calibration Optional calibration-assessment object.
- feature_manifest Optional feature-provenance manifest.
- external_validation Optional external-validation result.
- limitations Character vector describing model limitations.
- ethical_review Optional ethical-review information.

Value

A gp3ml_model_card object containing task, model, governance, evaluation, calibration, provenance, external-validation, and limitation metadata.

Examples

```

training_data <- data.frame(
  participant_id = rep(sprintf("P%02d", 1:12), each = 2),
  trial_id = sprintf("T%02d", 1:24),
  stimulus_id = rep(c("S01", "S02"), 12),
  fixation_duration = 180 + seq_len(24),
  pupil_change = sin(seq_len(24) / 3),
  stringsAsFactors = FALSE
)
training_data$quality_status <- factor(
  c(
    "pass", "review", "pass", "review", "review", "pass",
    "review", "pass", "pass", "review", "review", "pass",
    "review", "pass", "review", "pass", "pass", "review",
    "pass", "review", "review", "pass", "pass", "review"
  ),
  levels = c("pass", "review")
)
task <- declare_gazepoint_task(
  data = training_data,
  outcome = "quality_status",
  purpose = "Predict predefined recording-quality review status",
  task_type = "classification",
  unit_id = "trial_id",
  participant_id = "participant_id",
  stimulus_id = "stimulus_id",
  generalization_target = "new_participants",
  positive = "review"
)
model <- train_gazepoint_classifier(
  data = training_data,
  task = task,
  predictors = c("fixation_duration", "pupil_change"),
  engine = "glm",
  seed = 101L
)
card <- create_gazepoint_model_card(
  model = model,
  intended_use = paste(
    "Support manual review of predefined",
    "recording-quality status"
  ),
  limitations = "Synthetic example for documentation."
)
card

```

Description

Inner folds are constructed only from each outer analysis partition and preserve the declared participant/stimulus generalization target. The outer assessment partition is never used for inner preprocessing or tuning.

Usage

```
create_gazepoint_nested_folds(
  outer_folds,
  inner_v = 3L,
  inner_repeats = 1L,
  seed = 1L,
  continue_on_error = FALSE
)

## S3 method for class 'gp3ml_nested_folds'
print(x, ...)
```

Arguments

<code>outer_folds</code>	A validated gazepoint_group_folds object.
<code>inner_v</code>	Number of inner folds.
<code>inner_repeats</code>	Number of inner repeats.
<code>seed</code>	Base deterministic seed.
<code>continue_on_error</code>	Whether infeasible outer folds are retained as failures instead of stopping immediately.
<code>x</code>	An object returned by the corresponding gp3ml constructor, evaluator, summarizer, or validator.
<code>...</code>	Additional arguments passed to the print method.

Value

A gp3ml_nested_folds object.

Examples

```
data <- simulate_gazepoint_governed_data(16L, 4L, 1L, 303L)
predictors <- c("tracking_ratio", "blink_rate", "gaze_dispersion")
manifest <- create_gazepoint_synthetic_manifest("quality_status", predictors)
outer <- create_gazepoint_group_folds(
  data, "quality_status", predictors, manifest,
  "new_participants", "participant_id", "trial_id", "stimulus_id",
  v = 4L, repeats = 1L, seed = 303L
)
nested <- create_gazepoint_nested_folds(
  outer,
  inner_v = 3L,
```

```
    inner_repeats = 1L,  
    seed = 303L  
  )  
  nested
```

```
create_gazepoint_release_evidence
```

Create a release evidence manifest

Description

Create a release evidence manifest

Usage

```
create_gazepoint_release_evidence(  
  objects = list(),  
  files = character(),  
  version = "0.2.0",  
  notes = character()  
)  
  
## S3 method for class 'gp3ml_release_evidence'  
print(x, ...)
```

Arguments

objects	Named analysis objects to fingerprint.
files	Named file paths to checksum.
version	Intended future release version.
notes	Optional release notes.
x	An object returned by the corresponding gp3ml constructor, evaluator, summarizer, or validator.
...	Additional arguments passed to the print method.

Value

A gp3ml_release_evidence object.

```
create_gazepoint_release_model_card
```

Create a release-ready governed model card

Description

Extends the existing model-card structure with explicit model-selection, target-aligned uncertainty, nested-resampling, and transportability fields.

Usage

```
create_gazepoint_release_model_card(
  model,
  intended_use,
  evaluation = NULL,
  selection = NULL,
  uncertainty = NULL,
  calibration = NULL,
  feature_manifest = NULL,
  transportability = NULL,
  limitations,
  ethical_review = NULL,
  deployment_status = "research_review_only"
)

## S3 method for class 'gp3ml_release_model_card'
print(x, ...)
```

Arguments

<code>model</code>	Fitted governed model.
<code>intended_use</code>	Intended scientific use.
<code>evaluation</code>	Grouped or nested evaluation.
<code>selection</code>	Optional <code>gp3ml_model_selection</code> .
<code>uncertainty</code>	Optional target-aligned uncertainty object.
<code>calibration</code>	Optional calibration assessment.
<code>feature_manifest</code>	Optional feature manifest.
<code>transportability</code>	Optional transportability report.
<code>limitations</code>	Required limitations.
<code>ethical_review</code>	Optional ethical-review information.
<code>deployment_status</code>	Deployment status; defaults to research review only.

- x An object returned by the corresponding gp3ml constructor, evaluator, summarizer, or validator.
- ... Additional arguments passed to the print method.

Value

A gp3ml_release_model_card.

```
create_gazepoint_reproducibility_report
```

Create a reproducibility report

Description

Create a reproducibility report

Usage

```
create_gazepoint_reproducibility_report(
  objects = list(),
  data = NULL,
  seeds = list(),
  notes = character(),
  project_path = getwd()
)
```

Arguments

- objects Named objects to fingerprint.
- data Optional data frame to fingerprint.
- seeds Named list of deterministic seeds.
- notes Optional reproducibility notes.
- project_path Project directory recorded in the report.

Value

A gp3ml_reproducibility_report object containing runtime information, object and data fingerprints, seeds, Git metadata, notes, and prohibited uses.

Examples

```

example_data <- data.frame(
  trial_id = sprintf("T%02d", 1:6),
  fixation_duration = c(190, 205, 198, 214, 202, 220),
  stringsAsFactors = FALSE
)
report <- create_gazepoint_reproducibility_report(
  objects = list(
    fixation_values = example_data$fixation_duration
  ),
  data = example_data,
  seeds = list(example = 101L),
  notes = "Synthetic documentation example.",
  project_path = tempdir()
)
report

```

```
create_gazepoint_synthetic_manifest
```

Create a synthetic governed feature manifest

Description

Create a synthetic governed feature manifest

Usage

```

create_gazepoint_synthetic_manifest(
  outcome,
  predictors,
  participant_id = "participant_id",
  stimulus_id = "stimulus_id",
  trial_id = "trial_id"
)

```

Arguments

outcome	Name of the observed synthetic outcome.
predictors	Predictor names to declare.
participant_id	Participant identifier column.
stimulus_id	Stimulus identifier column.
trial_id	Trial identifier column.

Value

A gazepoint_feature_manifest produced by [create_gazepoint_feature_manifest\(\)](#).

Examples

```
create_gazepoint_synthetic_manifest(  
  outcome = "quality_status",  
  predictors = c("tracking_ratio", "blink_rate", "gaze_dispersion")  
)
```

```
create_gazepoint_synthetic_task
```

Create one of the governed synthetic demonstration tasks

Description

Create one of the governed synthetic demonstration tasks

Usage

```
create_gazepoint_synthetic_task(  
  data,  
  workflow = c("recording_quality", "assigned_condition", "observed_behavior",  
    "observed_duration"),  
  generalization_target = c("new_trials_known_participants", "new_participants",  
    "new_stimuli", "new_participants_and_new_stimuli")  
)
```

Arguments

data	Synthetic data from simulate_gazepoint_governed_data() .
workflow	Workflow name.
generalization_target	Declared generalization target.

Value

A governed gp3ml_task.

Examples

```
synthetic <- simulate_gazepoint_governed_data(12L, 4L, 1L, 101L)  
create_gazepoint_synthetic_task(  
  synthetic,  
  workflow = "recording_quality",  
  generalization_target = "new_participants"  
)
```

```
create_gazepoint_tuning_grid
```

Create an explicit governed tuning grid

Description

Candidate values are fully materialized before evaluation. No hidden metric, default ranking rule, or automatic winner is created.

Usage

```
create_gazepoint_tuning_grid(
  engine,
  engine_grid = list(),
  preprocessor_grid = list(),
  thresholds = 0.5,
  complexity = NA,
  interpretability = NA,
  labels = NULL
)

## S3 method for class 'gp3ml_tuning_grid'
print(x, ...)
```

Arguments

<code>engine</code>	One or more governed engine names.
<code>engine_grid</code>	Named list of engine-argument candidate values.
<code>preprocessor_grid</code>	Named list of preprocessing-argument candidate values.
<code>thresholds</code>	One or more explicit classification thresholds.
<code>complexity</code>	Optional complexity labels or numeric scores.
<code>interpretability</code>	Optional interpretability labels or numeric scores.
<code>labels</code>	Optional candidate labels.
<code>x</code>	An object returned by the corresponding gp3ml constructor, evaluator, summarizer, or validator.
<code>...</code>	Additional arguments passed to the print method.

Value

A `gp3ml_tuning_grid` with one row per explicit candidate.

Examples

```
grid <- create_gazepoint_tuning_grid(
  engine = "glm",
  preprocessor_grid = list(center = c(TRUE, FALSE), scale = TRUE),
  thresholds = c(0.4, 0.5),
  complexity = "low",
  interpretability = "high"
)
grid
```

```
create_gp3ml_governance_profile
      Create a governance-evidence profile
```

Description

Create a governance-evidence profile

Usage

```
create_gp3ml_governance_profile(
  evidence,
  framework = c("gp3ml-native", "NIST-AI-RMF-1.0", "ISO-23894-oriented",
    "ISO-42001-oriented")
)
```

Arguments

evidence	Named list of gp3ml evidence objects.
framework	Governance crosswalk.

Value

A gp3ml_governance_profile.

```
declare_gazepoint_analysis_plan
      Declare a frozen-analysis-plan contract
```

Description

Declare a frozen-analysis-plan contract

Usage

```

declare_gazepoint_analysis_plan(
  research_question,
  scientific_purpose,
  outcome,
  outcome_definition,
  predictors,
  generalization_target,
  grouping_variables = character(),
  eligible_population,
  exclusion_rules = character(),
  preprocessing_plan,
  candidate_models,
  primary_metric,
  secondary_metrics = character(),
  calibration_metric = NULL,
  uncertainty_method,
  threshold_policy = NULL,
  external_validation_required = FALSE,
  seed_strategy,
  prohibited_interpretations = gp3ml_prohibited_uses()
)

```

Arguments

<code>research_question</code>	Research question.
<code>scientific_purpose</code>	Explicit scientific purpose.
<code>outcome</code>	Outcome name.
<code>outcome_definition</code>	Operational definition of the observed outcome.
<code>predictors</code>	Predeclared predictors.
<code>generalization_target</code>	Intended generalization target.
<code>grouping_variables</code>	Grouping columns.
<code>eligible_population</code>	Eligibility statement.
<code>exclusion_rules</code>	Character vector of predeclared exclusions.
<code>preprocessing_plan</code>	Preprocessing plan.
<code>candidate_models</code>	Candidate model specifications or names.
<code>primary_metric</code>	Primary metric.

```

secondary_metrics      Secondary metrics.
calibration_metric      Calibration metric.
uncertainty_method      Uncertainty method.
threshold_policy        Threshold/decision policy.
external_validation_required
                        Whether independent validation is required.
seed_strategy           Deterministic seed strategy.
prohibited_interpretations
                        Character vector of prohibited interpretations.

```

Value

A mutable gp3ml_analysis_plan until locked.

```

declare_gazepoint_external_dataset
      Declare an external dataset and its independence status

```

Description

Declare an external dataset and its independence status

Usage

```

declare_gazepoint_external_dataset(
  data,
  label,
  independent,
  origin,
  collection_period = NULL,
  participant_id = "participant_id",
  stimulus_id = "stimulus_id",
  notes = character()
)

## S3 method for class 'gp3ml_external_dataset_declaration'
print(x, ...)

```

Arguments

data	Candidate external-validation data.
label	Dataset label.
independent	Explicit logical declaration of independence from model development and internal resampling.
origin	Human-readable origin or collection source.
collection_period	Optional collection period.
participant_id	Participant identifier column.
stimulus_id	Stimulus identifier column.
notes	Optional notes.
x	An object returned by the corresponding gp3ml constructor, evaluator, summarizer, or validator.
...	Additional arguments passed to the print method.

Value

A gp3ml_external_dataset_declaration.

Examples

```
external <- simulate_gazepoint_governed_data(8L, 4L, 1L, 505L)
declaration <- declare_gazepoint_external_dataset(
  external,
  label = "synthetic_external_site",
  independent = TRUE,
  origin = "Independent deterministic synthetic site"
)
declaration
```

```
declare_gazepoint_task
```

Declare a governed Gazepoint prediction task

Description

Declare a governed Gazepoint prediction task

Usage

```
declare_gazepoint_task(
  data,
  outcome,
  purpose,
  task_type = c("classification", "regression"),
  unit_id,
  participant_id = NULL,
  stimulus_id = NULL,
  generalization_target = c("new_trials_known_participants", "new_participants",
    "new_stimuli", "new_participants_and_new_stimuli", "external_validation"),
  positive = NULL,
  observed_outcome = TRUE,
  sensitive_outcome = FALSE
)
```

Arguments

<code>data</code>	A data frame containing the outcome and task identifiers.
<code>outcome</code>	Name of the explicitly observed outcome column.
<code>purpose</code>	One explicit scientific-purpose statement.
<code>task_type</code>	Either classification or regression.
<code>unit_id</code>	Column identifying the prediction unit.
<code>participant_id</code>	Optional participant-identifier column.
<code>stimulus_id</code>	Optional stimulus-identifier column.
<code>generalization_target</code>	The intended target of generalization.
<code>positive</code>	Positive outcome level for binary classification.
<code>observed_outcome</code>	Whether the outcome was directly observed.
<code>sensitive_outcome</code>	Whether the outcome is sensitive or prohibited.

Value

A governed `gp3ml_task` object describing the outcome, scientific purpose, prediction unit, grouping roles, task type, and generalization target.

Examples

```
example_data <- data.frame(
  participant_id = rep(sprintf("P%02d", 1:12), each = 2),
  trial_id = sprintf("T%02d", 1:24),
  stimulus_id = rep(c("S01", "S02"), 12),
  condition = rep(c("A", "B"), 12),
  fixation_duration = 180 + seq_len(24),
```

```

pupil_change = sin(seq_len(24) / 3),
stringsAsFactors = FALSE
)
example_data$quality_status <- factor(
  c(
    "pass", "review", "pass", "review", "review", "pass",
    "review", "pass", "pass", "review", "review", "pass",
    "review", "pass", "review", "pass", "pass", "review",
    "pass", "review", "review", "pass", "pass", "review"
  ),
  levels = c("pass", "review")
)
task <- declare_gazepoint_task(
  data = example_data,
  outcome = "quality_status",
  purpose = "Predict predefined recording-quality review status",
  task_type = "classification",
  unit_id = "trial_id",
  participant_id = "participant_id",
  stimulus_id = "stimulus_id",
  generalization_target = "new_participants",
  positive = "review"
)
task

```

diagnose_gazepoint_group_folds

Diagnose group-aware Gazepoint resampling folds

Description

Creates fold-size, repeat-level, grouping, assessment-coverage, outcome-balance, and exclusion diagnostics for an existing `gazepoint_group_folds` object.

Usage

```
diagnose_gazepoint_group_folds(x, imbalance_review = 1.5, imbalance_fail = 2)
```

Arguments

`x` A `gazepoint_group_folds` object.

`imbalance_review` Fold-size ratio above which diagnostics receive a review status.

`imbalance_fail` Fold-size ratio above which diagnostics receive a fail status.

Value

An object of class `gazepoint_fold_diagnostics`.

Examples

```

example_data <- expand.grid(
  participant_id = sprintf("P%02d", 1:6),
  stimulus_id = sprintf("S%02d", 1:4),
  repetition = 1:2,
  KEEP.OUT.ATTRS = FALSE,
  stringsAsFactors = FALSE
)
example_data$trial_id <- paste0(
  example_data$stimulus_id,
  "_T",
  example_data$repetition
)
participant_number <- as.integer(
  sub("P", "", example_data$participant_id)
)
stimulus_number <- as.integer(
  sub("S", "", example_data$stimulus_id)
)
example_data$outcome <- factor(
  ifelse(
    (participant_number + stimulus_number) %% 2L == 0L,
    "review",
    "pass"
  ),
  levels = c("pass", "review")
)
row_index <- seq_len(nrow(example_data))
example_data$fixation_duration <- 180 + row_index
example_data$pupil_change <- round(
  sin(row_index / 7),
  4
)
example_data$repetition <- NULL
manifest <- create_gazepoint_feature_manifest(
  features = c("fixation_duration", "pupil_change"),
  scientific_source = c(
    "Gazepoint fixation export",
    "Gazepoint pupil export"
  ),
  source_table = c("fixations", "pupil"),
  transformation = c(
    "Trial-level mean",
    "Trial-level change"
  ),
  availability_stage = "during_exposure",
  prediction_time_available = TRUE,
  preprocessing_scope = "none",
  fold_local_required = FALSE
)
folds <- create_gazepoint_group_folds(
  data = example_data,

```

```

outcome = "outcome",
predictors = c("fixation_duration", "pupil_change"),
feature_manifest = manifest,
generalization_target = "new_participants",
participant_id = "participant_id",
trial_id = "trial_id",
stimulus_id = "stimulus_id",
v = 3L,
repeats = 1L,
seed = 101L
)
diagnostics <- diagnose_gazepoint_group_folds(folds)
diagnostics

```

```
evaluate_external_validation
```

Evaluate an independent external-validation dataset

Description

Evaluate an independent external-validation dataset

Usage

```

evaluate_external_validation(
  model,
  external_data,
  label = "external",
  threshold = model$threshold,
  bootstrap = 200L,
  seed = 1L
)

```

Arguments

<code>model</code>	A fitted <code>gp3ml_model</code> object.
<code>external_data</code>	Independent external-validation data.
<code>label</code>	Label identifying the validation dataset.
<code>threshold</code>	Classification probability threshold.
<code>bootstrap</code>	Number of calibration bootstrap replicates.
<code>seed</code>	Deterministic random seed.

Value

A `gp3ml_external_validation` object containing external predictions, performance metrics, calibration results where applicable, predictor-shift diagnostics, a dataset fingerprint, and task metadata.

Examples

```

training_data <- data.frame(
  participant_id = rep(sprintf("P%02d", 1:12), each = 2),
  trial_id = sprintf("T%02d", 1:24),
  stimulus_id = rep(c("S01", "S02"), 12),
  fixation_duration = 180 + seq_len(24),
  pupil_change = sin(seq_len(24) / 3),
  stringsAsFactors = FALSE
)
training_data$quality_status <- factor(
  c(
    "pass", "review", "pass", "review", "review", "pass",
    "review", "pass", "pass", "review", "review", "pass",
    "review", "pass", "review", "pass", "pass", "review",
    "pass", "review", "review", "pass", "pass", "review"
  ),
  levels = c("pass", "review")
)
task <- declare_gazepoint_task(
  data = training_data,
  outcome = "quality_status",
  purpose = "Predict predefined recording-quality review status",
  task_type = "classification",
  unit_id = "trial_id",
  participant_id = "participant_id",
  stimulus_id = "stimulus_id",
  generalization_target = "new_participants",
  positive = "review"
)
model <- train_gazepoint_classifier(
  data = training_data,
  task = task,
  predictors = c("fixation_duration", "pupil_change"),
  engine = "glm",
  seed = 101L
)
external_data <- training_data
external_data$participant_id <- rep(
  sprintf("E%02d", 1:12),
  each = 2
)
external_data$trial_id <- sprintf("ET%02d", 1:24)
external_data$fixation_duration <-
  external_data$fixation_duration + 4
external_data$pupil_change <- cos(seq_len(24) / 4)
validation <- evaluate_external_validation(
  model = model,
  external_data = external_data,
  label = "synthetic_external",
  bootstrap = 10L,
  seed = 101L
)

```

validation

evaluate_gazepoint_external_transportability

Evaluate external transportability and validation status

Description

An internal holdout or a dataset explicitly declared non-independent is labelled `not_externally_validated`; it cannot generate an external- validation claim.

Usage

```
evaluate_gazepoint_external_transportability(
  model,
  development_data,
  external_data = NULL,
  declaration = NULL,
  development_evaluation = NULL,
  threshold = model$threshold,
  bootstrap = 200L,
  seed = 1L
)

## S3 method for class 'gp3ml_transportability_report'
print(x, ...)
```

Arguments

<code>model</code>	Fitted governed model.
<code>development_data</code>	Data used to characterize development schema and group coverage.
<code>external_data</code>	Candidate external data. May be <code>NULL</code> to create an explicit not-validated status.
<code>declaration</code>	External dataset declaration. Required when external data are supplied.
<code>development_evaluation</code>	Optional grouped development evaluation.
<code>threshold</code>	Classification threshold.
<code>bootstrap</code>	Calibration bootstrap replicates.
<code>seed</code>	Deterministic seed.
<code>x</code>	An object returned by the corresponding <code>gp3ml</code> constructor, evaluator, summarizer, or validator.
<code>...</code>	Additional arguments passed to the print method.

Value

A `gp3ml_transportability_report` object.

evaluate_gazepoint_feature_stability
Evaluate leave-one-feature-out stability

Description

Evaluate leave-one-feature-out stability

Usage

```
evaluate_gazepoint_feature_stability(features, evaluator, ...)
```

Arguments

features	Predictor names.
evaluator	Function called as evaluator(excluded_feature = feature, ...).
...	Additional evaluator arguments.

Value

A gp3ml_stability_evaluation.

evaluate_gazepoint_group_folds
Evaluate a governed model specification across materialized grouped folds

Description

Fits preprocessing and the requested model only on each fold's analysis partition, predicts only on the corresponding assessment partition, retains excluded rows, and records fold-level metrics, leakage audits, warnings, and failures. Row-level predictions are never relabelled as participant- or stimulus-level estimates.

Usage

```
evaluate_gazepoint_group_folds(  
  folds,  
  task,  
  predictors = NULL,  
  engine = NULL,  
  preprocessor_args = list(),  
  engine_args = list(),  
  threshold = 0.5,  
  seed = 1L,
```

```

    assess_calibration = FALSE,
    calibration_bins = 10L,
    calibration_bootstrap = 0L,
    keep_models = FALSE,
    continue_on_error = TRUE
  )

  ## S3 method for class 'gp3ml_resample_evaluation'
  print(x, ...)

```

Arguments

<code>folds</code>	A mature <code>gazepoint_group_folds</code> object containing materialized folds under <code>folds\$folds</code> .
<code>task</code>	A governed <code>gp3ml_task</code> compatible with the fold metadata.
<code>predictors</code>	Optional predictor names. Defaults to the fold metadata.
<code>engine</code>	Model engine name or governed custom engine.
<code>preprocessor_args</code>	Arguments passed to <code>fit_gazepoint_preprocessor()</code> .
<code>engine_args</code>	Arguments passed to <code>fit_gazepoint_model()</code> .
<code>threshold</code>	Classification threshold.
<code>seed</code>	Base deterministic seed.
<code>assess_calibration</code>	Whether to calculate assessment-fold calibration summaries for classification tasks.
<code>calibration_bins</code>	Number of reliability bins.
<code>calibration_bootstrap</code>	Calibration bootstrap replicates. Use zero in fast smoke tests.
<code>keep_models</code>	Whether fitted fold models are retained.
<code>continue_on_error</code>	Whether later folds continue after a failed fold.
<code>x</code>	An object returned by the corresponding <code>gp3ml</code> constructor, evaluator, summarizer, or validator.
<code>...</code>	Additional arguments passed to the print method.

Value

A `gp3ml_resample_evaluation` object.

Examples

```

data <- simulate_gazepoint_governed_data(12L, 4L, 1L, 101L)
predictors <- c("tracking_ratio", "blink_rate", "gaze_dispersion")
manifest <- create_gazepoint_synthetic_manifest("quality_status", predictors)
folds <- create_gazepoint_group_folds(

```

```

    data = data,
    outcome = "quality_status",
    predictors = predictors,
    feature_manifest = manifest,
    generalization_target = "new_participants",
    participant_id = "participant_id",
    trial_id = "trial_id",
    stimulus_id = "stimulus_id",
    v = 3L,
    repeats = 1L,
    seed = 101L
  )
task <- create_gazepoint_synthetic_task(
  data,
  "recording_quality",
  "new_participants"
)
evaluation <- evaluate_gazepoint_group_folds(
  folds,
  task,
  predictors = predictors,
  engine = "glm",
  seed = 101L
)
evaluation

```

```
evaluate_gazepoint_missingness_sensitivity
```

Evaluate named missingness-sensitivity scenarios

Description

Evaluate named missingness-sensitivity scenarios

Usage

```
evaluate_gazepoint_missingness_sensitivity(scenarios, evaluator, ...)
```

Arguments

scenarios	Named list of scenario objects.
evaluator	Function called as <code>evaluator(scenario = scenario, name = name, ...)</code> .
...	Additional evaluator arguments.

Value

A `gp3ml_stability_evaluation`.

```
evaluate_gazepoint_nested_resampling
```

Evaluate nested grouped resampling with inner governed tuning

Description

Evaluate nested grouped resampling with inner governed tuning

Usage

```
evaluate_gazepoint_nested_resampling(
  nested_folds,
  task,
  tuning_grid,
  selection_metric,
  direction,
  predictors = NULL,
  minimum_success_prop = 0.8,
  tie_breakers = NULL,
  selection_rationale = .gp3ml_nested_selection_rationale_default,
  seed = 1L,
  keep_models = FALSE,
  continue_on_error = TRUE
)

## S3 method for class 'gp3ml_nested_evaluation'
print(x, ...)
```

Arguments

nested_folds	A gp3ml_nested_folds object.
task	Governed task.
tuning_grid	Explicit tuning grid.
selection_metric	Explicit inner selection metric.
direction	Explicit selection direction.
predictors	Optional predictors.
minimum_success_prop	Minimum inner-fold success proportion.
tie_breakers	Optional secondary metrics.
selection_rationale	Human rationale recorded for each outer fold.
seed	Base deterministic seed.
keep_models	Whether outer fitted models are retained.

continue_on_error	Whether failed outer folds remain in the result.
x	An object returned by the corresponding gp3ml constructor, evaluator, summarizer, or validator.
...	Additional arguments passed to the print method.

Value

A gp3ml_nested_evaluation object retaining inner tuning results, selections, outer predictions, metrics, and failures.

evaluate_gazepoint_seed_stability
Evaluate seed stability

Description

Evaluate seed stability

Usage

```
evaluate_gazepoint_seed_stability(seeds, evaluator, ...)
```

Arguments

seeds	Integer seeds.
evaluator	Function called as evaluator(seed = seed, ...).
...	Additional evaluator arguments.

Value

A gp3ml_stability_evaluation.

evaluate_gazepoint_threshold_stability
Evaluate threshold stability around the optimum

Description

Evaluate threshold stability around the optimum

Usage

```
evaluate_gazepoint_threshold_stability(
  evaluation,
  metric,
  direction = c("maximize", "minimize"),
  tolerance = 0.02
)
```

Arguments

evaluation	Threshold evaluation.
metric	Metric.
direction	Optimization direction.
tolerance	Fractional tolerance from the optimum.

Value

A gp3ml_threshold_stability.

evaluate_gazepoint_thresholds

Evaluate explicit classification thresholds

Description

Evaluate explicit classification thresholds

Usage

```
evaluate_gazepoint_thresholds(
  truth,
  probability,
  positive,
  thresholds,
  cost_false_positive = 1,
  cost_false_negative = 1
)
```

Arguments

truth	Observed binary outcome.
probability	Probability of the positive class.
positive	Positive class label.
thresholds	Explicit candidate thresholds.

cost_false_positive
False-positive cost.

cost_false_negative
False-negative cost.

Value

A gp3ml_threshold_evaluation.

fit_gazepoint_calibrator
Fit a probability calibrator

Description

Fit a probability calibrator

Usage

```
fit_gazepoint_calibrator(
  truth,
  probability,
  positive = NULL,
  method = c("platt", "isotonic")
)
```

Arguments

truth	Observed binary outcome values.
probability	Uncalibrated positive-class probabilities.
positive	Label representing the positive class.
method	Calibration method: Platt scaling or isotonic regression.

Value

A fitted gp3ml_calibrator object containing the calibration method, fitted model, and outcome labels.

Examples

```
truth <- factor(
  rep(c("pass", "review"), 6),
  levels = c("pass", "review")
)
probability <- c(
  0.20, 0.70, 0.60, 0.55, 0.30, 0.80,
  0.65, 0.45, 0.40, 0.75, 0.50, 0.60
)
```

```

)
calibrator <- fit_gazepoint_calibrator(
  truth = truth,
  probability = probability,
  positive = "review",
  method = "platt"
)
calibrator

```

fit_gazepoint_conformal

Fit target-aware split-conformal calibration

Description

This function provides conservative split-conformal calibration for explicitly observed regression or binary classification outcomes. When a grouped calibration unit is supplied, row scores are aggregated to the maximum score within each calibration unit before the conformal quantile is estimated. This records and respects the calibration unit but does not claim distribution-free coverage under arbitrary dependence.

Usage

```

fit_gazepoint_conformal(
  truth,
  prediction = NULL,
  probability = NULL,
  task_type = c("regression", "classification"),
  positive = NULL,
  level = 0.9,
  calibration_unit = c("observation", "participant", "stimulus", "participant_stimulus"),
  unit = NULL,
  generalization_target
)

```

Arguments

truth	Observed calibration outcomes.
prediction	Numeric predictions for regression.
probability	Positive-class probabilities for classification.
task_type	"regression" or "classification".
positive	Positive class label for classification.
level	Nominal coverage level.
calibration_unit	Calibration unit.
unit	Optional group identifier for grouped calibration.
generalization_target	Declared generalization target.

Value

A gp3ml_conformal_fit.

fit_gazepoint_deep_model

Fit an optional governed deep-learning model through keras3

Description

Fit an optional governed deep-learning model through keras3

Usage

```
fit_gazepoint_deep_model(
  data,
  task,
  predictors = NULL,
  preprocessor = NULL,
  hidden_units = c(64L, 32L),
  dropout = 0.2,
  epochs = 50L,
  batch_size = 32L,
  validation_split = 0.2,
  optimizer = "adam",
  seed = 1L,
  verbose = 0L
)
```

Arguments

data	Analysis data used to fit the network.
task	A governed gp3ml_task object.
predictors	Optional character vector of predictor columns.
preprocessor	Optional fitted preprocessing object.
hidden_units	Integer vector of hidden-layer sizes.
dropout	Dropout proportion applied after hidden layers.
epochs	Number of training epochs.
batch_size	Training batch size.
validation_split	Proportion reserved for internal validation.
optimizer	Keras optimizer name or object.
seed	Deterministic random seed.
verbose	Keras training verbosity.

Value

A governed gp3ml_model object containing the fitted keras3 model, training history, preprocessing object, task contract, and training metadata.

Examples

```
example_data <- data.frame(
  participant_id = rep(sprintf("P%02d", 1:12), each = 2),
  trial_id = sprintf("T%02d", 1:24),
  stimulus_id = rep(c("S01", "S02"), 12),
  condition = rep(c("A", "B"), 12),
  fixation_duration = 180 + seq_len(24),
  pupil_change = sin(seq_len(24) / 3),
  stringsAsFactors = FALSE
)
example_data$quality_status <- factor(
  c(
    "pass", "review", "pass", "review", "review", "pass",
    "review", "pass", "pass", "review", "review", "pass",
    "review", "pass", "review", "pass", "pass", "review",
    "pass", "review", "review", "pass", "pass", "review"
  ),
  levels = c("pass", "review")
)
task <- declare_gazepoint_task(
  data = example_data,
  outcome = "quality_status",
  purpose = "Predict predefined recording-quality review status",
  task_type = "classification",
  unit_id = "trial_id",
  participant_id = "participant_id",
  stimulus_id = "stimulus_id",
  generalization_target = "new_participants",
  positive = "review"
)
deep_model <- fit_gazepoint_deep_model(
  data = example_data,
  task = task,
  predictors = c("fixation_duration", "pupil_change"),
  hidden_units = 4L,
  dropout = 0,
  epochs = 1L,
  batch_size = 8L,
  validation_split = 0,
  seed = 101L,
  verbose = 0L
)
deep_model
```

fit_gazepoint_model	<i>Fit a governed Gazepoint model</i>
---------------------	---------------------------------------

Description

Fit a governed Gazepoint model

Usage

```
fit_gazepoint_model(
  data,
  task,
  predictors = NULL,
  engine = NULL,
  preprocessor = NULL,
  preprocessor_args = list(),
  engine_args = list(),
  seed = 1L,
  threshold = 0.5
)
```

Arguments

data	Analysis data used to fit the model.
task	A governed gp3ml_task object.
predictors	Optional character vector of predictor columns.
engine	Engine name or controlled custom-engine object.
preprocessor	Optional fitted preprocessing object.
preprocessor_args	Arguments passed to preprocessing fitting.
engine_args	Arguments passed to the model engine.
seed	Deterministic random seed.
threshold	Classification probability threshold.

Value

A governed gp3ml_model object containing the fitted engine, preprocessing object, task contract, predictors, and training metadata.

Examples

```
example_data <- data.frame(
  participant_id = rep(sprintf("P%02d", 1:12), each = 2),
  trial_id = sprintf("T%02d", 1:24),
  stimulus_id = rep(c("S01", "S02"), 12),
```

```

condition = rep(c("A", "B"), 12),
fixation_duration = 180 + seq_len(24),
pupil_change = sin(seq_len(24) / 3),
stringsAsFactors = FALSE
)
example_data$quality_status <- factor(
  c(
    "pass", "review", "pass", "review", "review", "pass",
    "review", "pass", "pass", "review", "review", "pass",
    "review", "pass", "review", "pass", "pass", "review",
    "pass", "review", "review", "pass", "pass", "review"
  ),
  levels = c("pass", "review")
)
task <- declare_gazepoint_task(
  data = example_data,
  outcome = "quality_status",
  purpose = "Predict predefined recording-quality review status",
  task_type = "classification",
  unit_id = "trial_id",
  participant_id = "participant_id",
  stimulus_id = "stimulus_id",
  generalization_target = "new-participants",
  positive = "review"
)
model <- fit_gazepoint_model(
  data = example_data,
  task = task,
  predictors = c("fixation_duration", "pupil_change"),
  engine = "glm",
  seed = 101L
)
model

```

fit_gazepoint_preprocessor

Fit a fold-local preprocessing engine

Description

Fit a fold-local preprocessing engine

Usage

```

fit_gazepoint_preprocessor(
  data,
  predictors,
  numeric_imputation = c("median", "mean"),
  center = TRUE,
  scale = TRUE,

```

```

    novel_level = c("other", "error"),
    remove_zero_variance = TRUE
  )

```

Arguments

<code>data</code>	Analysis data used to estimate preprocessing parameters.
<code>predictors</code>	Character vector naming predictor columns.
<code>numeric_imputation</code>	Numeric imputation method.
<code>center</code>	Whether numeric model columns should be centered.
<code>scale</code>	Whether numeric model columns should be scaled.
<code>novel_level</code>	How novel categorical levels should be handled.
<code>remove_zero_variance</code>	Whether zero-variance columns are removed.

Value

A fitted `gp3ml_preprocessor` object containing analysis-partition imputation values, factor levels, model columns, centering values, and scaling values.

Examples

```

example_data <- data.frame(
  participant_id = rep(sprintf("P%02d", 1:12), each = 2),
  trial_id = rep(sprintf("T%02d", 1:24),
  stimulus_id = rep(c("S01", "S02"), 12),
  condition = rep(c("A", "B"), 12),
  fixation_duration = 180 + seq_len(24),
  pupil_change = sin(seq_len(24) / 3),
  stringsAsFactors = FALSE
)
example_data$quality_status <- factor(
  c(
    "pass", "review", "pass", "review", "review", "pass",
    "review", "pass", "pass", "review", "review", "pass",
    "review", "pass", "review", "pass", "pass", "review",
    "pass", "review", "review", "pass", "pass", "review"
  ),
  levels = c("pass", "review")
)
preprocessor <- fit_gazepoint_preprocessor(
  data = example_data,
  predictors = c(
    "fixation_duration",
    "pupil_change",
    "condition"
  )
)
preprocessor

```

```
gazepoint_classification_metrics
```

Binary classification metrics

Description

Binary classification metrics

Usage

```
gazepoint_classification_metrics(
  truth,
  probability,
  predicted = NULL,
  positive = NULL,
  threshold = 0.5
)
```

Arguments

<code>truth</code>	Observed binary outcome values.
<code>probability</code>	Predicted positive-class probabilities.
<code>predicted</code>	Optional predicted classes.
<code>positive</code>	Label representing the positive class.
<code>threshold</code>	Probability threshold used for class predictions.

Value

A one-row data frame containing the sample size, threshold, class-performance measures, discrimination metrics, Brier score, and log loss.

Examples

```
truth <- factor(
  rep(c("pass", "review"), 6),
  levels = c("pass", "review")
)
probability <- c(
  0.20, 0.70, 0.60, 0.55, 0.30, 0.80,
  0.65, 0.45, 0.40, 0.75, 0.50, 0.60
)
predicted <- factor(
  ifelse(probability >= 0.5, "review", "pass"),
  levels = levels(truth)
)
gazepoint_classification_metrics(
  truth = truth,
```

```

    probability = probability,
    predicted = predicted,
    positive = "review"
  )

```

```

gazeptoint_performance_metrics
      Task-aware performance metrics

```

Description

Task-aware performance metrics

Usage

```

gazeptoint_performance_metrics(
  task,
  truth,
  prediction = NULL,
  probability = NULL,
  threshold = 0.5
)

```

Arguments

<code>task</code>	A governed gp3ml_task object.
<code>truth</code>	Observed outcome values.
<code>prediction</code>	Predicted classes or numeric values.
<code>probability</code>	Predicted positive-class probabilities.
<code>threshold</code>	Probability threshold for classification.

Value

A one-row data frame of classification or regression metrics selected according to the governed task type.

Examples

```

example_data <- data.frame(
  participant_id = rep(sprintf("P%02d", 1:12), each = 2),
  trial_id = rep(sprintf("T%02d", 1:24),
  stimulus_id = rep(c("S01", "S02"), 12),
  condition = rep(c("A", "B"), 12),
  fixation_duration = 180 + seq_len(24),
  pupil_change = sin(seq_len(24) / 3),
  stringsAsFactors = FALSE
)

```

```

example_data$quality_status <- factor(
  c(
    "pass", "review", "pass", "review", "review", "pass",
    "review", "pass", "pass", "review", "review", "pass",
    "review", "pass", "review", "pass", "pass", "review",
    "pass", "review", "review", "pass", "pass", "review"
  ),
  levels = c("pass", "review")
)
task <- declare_gazeptoint_task(
  data = example_data,
  outcome = "quality_status",
  purpose = "Predict predefined recording-quality review status",
  task_type = "classification",
  unit_id = "trial_id",
  participant_id = "participant_id",
  stimulus_id = "stimulus_id",
  generalization_target = "new_participants",
  positive = "review"
)
probability <- seq(
  0.20,
  0.80,
  length.out = nrow(example_data)
)
predicted <- factor(
  ifelse(probability >= 0.5, "review", "pass"),
  levels = levels(example_data$quality_status)
)
gazeptoint_performance_metrics(
  task = task,
  truth = example_data$quality_status,
  prediction = predicted,
  probability = probability
)

```

gazeptoint_regression_metrics

Regression metrics

Description

Regression metrics

Usage

```
gazeptoint_regression_metrics(truth, prediction)
```

Arguments

truth	Observed numeric outcome values.
prediction	Predicted numeric outcome values.

Value

A one-row data frame containing the sample size, RMSE, MAE, R-squared value, and prediction correlation.

Examples

```
truth <- c(1.0, 2.0, 3.0, 4.0, 5.0)
prediction <- c(1.1, 1.8, 3.2, 3.9, 4.8)
gazeport_regression_metrics(truth, prediction)
```

gp3ml_api_contracts	<i>gp3ml public API contracts</i>
---------------------	-----------------------------------

Description

Returns the package's explicit compatibility contract for the public API. APIs classified as stable in version 0.2.0 remain stable throughout the 0.3.x line. New APIs introduced by the current development milestone are marked experimental until promoted by a later release decision.

Usage

```
gp3ml_api_contracts()
```

Value

A gp3ml_api_contract_registry object.

gp3ml_available_engines	<i>List available model engines</i>
-------------------------	-------------------------------------

Description

List available model engines

Usage

```
gp3ml_available_engines()
```

Value

A data frame listing supported model-engine names and whether each optional engine is currently available.

Examples

```
gp3ml_available_engines()
```

```
gp3ml_engine_capabilities
```

Audit gp3ml model-engine capabilities

Description

Audit gp3ml model-engine capabilities

Usage

```
gp3ml_engine_capabilities(check_keras_backend = FALSE)
```

Arguments

check_keras_backend

Whether to query the configured Keras backend.

Value

A gp3ml_engine_capabilities data frame.

```
gp3ml_interop_contracts
```

Cross-package interoperability contracts

Description

Describes a lightweight handoff boundary. Upstream packages remain responsible for their own importing, cleaning, feature derivation, signal processing, sequence processing, and quality control. gp3ml receives already prepared observed variables together with explicit provenance.

Usage

```
gp3ml_interop_contracts()
```

Value

A data frame describing supported handoff sources and responsibilities.

gp3ml_object_schema	<i>Describe the schema of a gp3ml object</i>
---------------------	--

Description

Describe the schema of a gp3ml object

Usage

```
gp3ml_object_schema(x, recursive = FALSE)
```

Arguments

x	Object to inspect.
recursive	Whether to include one level of nested named-list components.

Value

A data frame describing component names, classes, storage types, lengths, and dimensions.

gp3ml_prohibited_uses	<i>Prohibited gp3ml uses</i>
-----------------------	------------------------------

Description

Prohibited gp3ml uses

Usage

```
gp3ml_prohibited_uses()
```

Value

A character vector of prohibited use descriptions.

Examples

```
gp3ml_prohibited_uses()
```

integrate_black_box_model

Integrate a controlled black-box model engine

Description

Integrate a controlled black-box model engine

Usage

```
integrate_black_box_model(
  name,
  fit_fun,
  predict_fun,
  supports = c("classification", "regression"),
  probability = TRUE,
  metadata = list(),
  safety_declaration
)
```

Arguments

name	Unique name for the custom engine.
fit_fun	Function that fits the custom engine.
predict_fun	Function that generates predictions.
supports	Task types supported by the engine.
probability	Whether classification probabilities are supported.
metadata	Optional engine metadata.
safety_declaration	Named logical safety declarations.

Value

A controlled `gp3ml_engine` object containing the custom fit and prediction functions, supported task types, metadata, and explicit safety declarations.

Examples

```
custom_fit <- function(x, y, task, args) {
  training_data <- data.frame(
    .outcome = y,
    x,
    check.names = FALSE
  )
  stats::glm(
    .outcome ~ .,

```

```

      data = training_data,
      family = stats::binomial()
    )
  }
  custom_predict <- function(fit, newdata, type, task, ...) {
    as.numeric(stats::predict(
      fit,
      newdata = as.data.frame(newdata),
      type = "response"
    ))
  }
  engine <- integrate_black_box_model(
    name = "custom_glm",
    fit_fun = custom_fit,
    predict_fun = custom_predict,
    supports = "classification",
    probability = TRUE,
    safety_declaration = list(
      prohibited_uses_acknowledged = TRUE,
      prediction_time_inputs_only = TRUE,
      group_aware_evaluation_required = TRUE
    )
  )
  engine$name
  engine$supports
  engine$probability
  engine$safety_declaration

```

lock_gazepoint_analysis_plan

Lock an analysis plan using SHA-256

Description

Lock an analysis plan using SHA-256

Usage

```
lock_gazepoint_analysis_plan(plan, plan_id = NULL, locked_at = Sys.time())
```

Arguments

plan	A valid unlocked plan.
plan_id	Optional stable identifier.
locked_at	Optional lock time.

Value

A locked gp3ml_analysis_plan.

`normalize_gazepoint_artifact_text`*Normalize volatile text in generated research artifacts*

Description

Normalize volatile text in generated research artifacts

Usage

```
normalize_gazepoint_artifact_text(x, project_path = NULL)
```

Arguments

`x` Character vector.
`project_path` Optional project path to replace by <PROJECT>.

Value

Character vector with volatile runtime fragments normalized.

`plot.gp3ml_abstention_audit`*Plot abstention audit*

Description

Plot abstention audit

Usage

```
## S3 method for class 'gp3ml_abstention_audit'  
plot(x, ...)
```

Arguments

`x` A gp3ml_abstention_audit.
`...` Additional arguments passed to `graphics::barplot()`.

```
plot.gp3ml_api_stability_audit
```

Plot a gp3ml API stability audit

Description

Plot a gp3ml API stability audit

Usage

```
## S3 method for class 'gp3ml_api_stability_audit'  
plot(x, ...)
```

Arguments

x	A gp3ml_api_stability_audit.
...	Additional graphical parameters.

```
plot.gp3ml_conformal_coverage
```

Plot conformal coverage

Description

Plot conformal coverage

Usage

```
## S3 method for class 'gp3ml_conformal_coverage'  
plot(x, ...)
```

Arguments

x	A gp3ml_conformal_coverage.
...	Additional arguments passed to graphics::barplot().

```
plot.gp3ml_dataset_shift_audit
```

Plot a dataset shift audit

Description

Plot a dataset shift audit

Usage

```
## S3 method for class 'gp3ml_dataset_shift_audit'  
plot(x, ...)
```

Arguments

x	A gp3ml_dataset_shift_audit.
...	Additional arguments passed to graphics::barplot().

```
plot.gp3ml_engine_capabilities
```

Plot gp3ml engine availability

Description

Plot gp3ml engine availability

Usage

```
## S3 method for class 'gp3ml_engine_capabilities'  
plot(x, ...)
```

Arguments

x	Engine-capability table.
...	Additional graphical parameters.

```
plot.gp3ml_environment_comparison
```

Plot an environment comparison

Description

Plot an environment comparison

Usage

```
## S3 method for class 'gp3ml_environment_comparison'
plot(x, ...)
```

Arguments

x	An environment comparison.
...	Additional arguments passed to <code>graphics::barplot()</code> .

```
plot.gp3ml_governance_profile_audit
```

Plot a governance profile audit

Description

Plot a governance profile audit

Usage

```
## S3 method for class 'gp3ml_governance_profile_audit'
plot(x, ...)
```

Arguments

x	Governance profile audit.
...	Additional arguments passed to <code>graphics::barplot()</code> .

`plot.gp3ml_handoff_validation`*Plot handoff validation checks*

Description

Plot handoff validation checks

Usage

```
## S3 method for class 'gp3ml_handoff_validation'  
plot(x, ...)
```

Arguments

x	A gp3ml_handoff_validation.
...	Additional graphical parameters.

`plot.gp3ml_model_artifact_validation`*Plot model-artifact validation*

Description

Plot model-artifact validation

Usage

```
## S3 method for class 'gp3ml_model_artifact_validation'  
plot(x, ...)
```

Arguments

x	A gp3ml_model_artifact_validation.
...	Additional arguments passed to <code>graphics::barplot()</code> .

```
plot.gp3ml_model_robustness_audit
```

Plot a robustness audit

Description

Plot a robustness audit

Usage

```
## S3 method for class 'gp3ml_model_robustness_audit'
plot(x, ...)
```

Arguments

x	A gp3ml_model_robustness_audit.
...	Additional arguments passed to graphics::barplot().

```
plot.gp3ml_plan_deviation_audit
```

Plot analysis-plan deviations

Description

Plot analysis-plan deviations

Usage

```
## S3 method for class 'gp3ml_plan_deviation_audit'
plot(x, ...)
```

Arguments

x	A gp3ml_plan_deviation_audit.
...	Additional arguments passed to graphics::barplot().

```
plot.gp3ml_release_checksum_validation
```

Plot release checksum validation

Description

Plot release checksum validation

Usage

```
## S3 method for class 'gp3ml_release_checksum_validation'
plot(x, ...)
```

Arguments

x	Release checksum validation.
...	Additional arguments passed to <code>graphics::barplot()</code> .

```
plot.gp3ml_reproducibility_audit
```

Plot a reproducibility-hardening audit

Description

Plot a reproducibility-hardening audit

Usage

```
## S3 method for class 'gp3ml_reproducibility_audit'
plot(x, ...)
```

Arguments

x	A <code>gp3ml_reproducibility_audit</code> .
...	Additional graphical parameters.

```
plot.gp3ml_research_bundle_validation
```

Plot research-bundle validation

Description

Plot research-bundle validation

Usage

```
## S3 method for class 'gp3ml_research_bundle_validation'
plot(x, ...)
```

Arguments

x	A gp3ml_research_bundle_validation.
...	Additional graphical parameters.

```
plot.gp3ml_ro_crate_validation
```

Plot RO-Crate validation

Description

Plot RO-Crate validation

Usage

```
## S3 method for class 'gp3ml_ro_crate_validation'
plot(x, ...)
```

Arguments

x	A gp3ml_ro_crate_validation.
...	Additional arguments passed to graphics::barplot().

```
plot.gp3ml_threshold_evaluation
```

Plot threshold evaluation

Description

Plot threshold evaluation

Usage

```
## S3 method for class 'gp3ml_threshold_evaluation'
plot(x, metric = "balanced_accuracy", ...)
```

Arguments

x	A gp3ml_threshold_evaluation.
metric	Metric to plot.
...	Additional arguments passed to graphics::plot().

```
predict.gp3ml_model
```

Predict from a gp3ml model

Description

Predict from a gp3ml model

Usage

```
## S3 method for class 'gp3ml_model'
predict(
  object,
  newdata,
  type = c("response", "probability", "class", "link"),
  ...
)
```

Arguments

object	A fitted gp3ml_model object.
newdata	New data containing the required predictors.
type	Requested prediction type.
...	Additional arguments passed to custom prediction methods.

Value

For classification with `type = "class"`, a factor of predicted classes. Otherwise, a numeric vector of probabilities, link-scale values, or regression predictions.

Examples

```
example_data <- data.frame(
  participant_id = rep(sprintf("P%02d", 1:12), each = 2),
  trial_id = sprintf("T%02d", 1:24),
  stimulus_id = rep(c("S01", "S02"), 12),
  condition = rep(c("A", "B"), 12),
  fixation_duration = 180 + seq_len(24),
  pupil_change = sin(seq_len(24) / 3),
  stringsAsFactors = FALSE
)
example_data$quality_status <- factor(
  c(
    "pass", "review", "pass", "review", "review", "pass",
    "review", "pass", "pass", "review", "review", "pass",
    "review", "pass", "review", "pass", "pass", "review",
    "pass", "review", "review", "pass", "pass", "review"
  ),
  levels = c("pass", "review")
)
task <- declare_gazepoint_task(
  data = example_data,
  outcome = "quality_status",
  purpose = "Predict predefined recording-quality review status",
  task_type = "classification",
  unit_id = "trial_id",
  participant_id = "participant_id",
  stimulus_id = "stimulus_id",
  generalization_target = "new_participants",
  positive = "review"
)
model <- train_gazepoint_classifier(
  data = example_data,
  task = task,
  predictors = c("fixation_duration", "pupil_change"),
  engine = "glm",
  seed = 101L
)
probability <- predict(
  model,
  example_data,
  type = "probability"
)
predicted_class <- predict(
  model,
  example_data,
  type = "class"
)
```

```
head(probability)
head(predicted_class)
```

predict_gazepoint_interval
<i>Predict conformal regression intervals</i>

Description

Predict conformal regression intervals

Usage

```
predict_gazepoint_interval(object, prediction)
```

Arguments

- object A regression gp3ml_conformal_fit.
- prediction Point predictions.

Value

Data frame with point prediction, lower, and upper limits.

predict_gazepoint_set	<i>Predict binary conformal prediction sets</i>
-----------------------	---

Description

Predict binary conformal prediction sets

Usage

```
predict_gazepoint_set(object, probability)
```

Arguments

- object A classification gp3ml_conformal_fit.
- probability Positive-class probabilities.

Value

A data frame containing set membership and a readable set label.

```
print.gazepoint_feature_manifest_validation
```

Print feature-manifest validation

Description

Print feature-manifest validation

Usage

```
## S3 method for class 'gazepoint_feature_manifest_validation'  
print(x, ...)
```

Arguments

x	An object returned by <code>validate_gazepoint_feature_manifest()</code> .
...	Additional arguments, currently unused.

Value

x, invisibly.

Examples

```
manifest <- create_gazepoint_feature_manifest(  
  features = "fixation_duration",  
  scientific_source = "Gazepoint fixation export",  
  source_table = "fixations",  
  transformation = "Trial-level mean",  
  availability_stage = "during_exposure",  
  prediction_time_available = TRUE,  
  preprocessing_scope = "none",  
  fold_local_required = FALSE  
)  
validation <- validate_gazepoint_feature_manifest(manifest)  
print(validation)
```

```
print.gazepoint_fold_diagnostics
```

Print Gazepoint fold diagnostics

Description

Print Gazepoint fold diagnostics

Usage

```
## S3 method for class 'gazepoint_fold_diagnostics'
print(x, ...)
```

Arguments

x A *gazepoint_fold_diagnostics* object.

... Additional arguments, currently unused.

Value

x, invisibly.

Examples

```
example_data <- expand.grid(
  participant_id = sprintf("P%02d", 1:6),
  stimulus_id = sprintf("S%02d", 1:4),
  repetition = 1:2,
  KEEP.OUT.ATTRS = FALSE,
  stringsAsFactors = FALSE
)
example_data$trial_id <- paste0(
  example_data$stimulus_id,
  "_T",
  example_data$repetition
)
participant_number <- as.integer(
  sub("P", "", example_data$participant_id)
)
stimulus_number <- as.integer(
  sub("S", "", example_data$stimulus_id)
)
example_data$outcome <- factor(
  ifelse(
    (participant_number + stimulus_number) %% 2L == 0L,
    "review",
    "pass"
  ),
  levels = c("pass", "review")
)
row_index <- seq_len(nrow(example_data))
example_data$fixation_duration <- 180 + row_index
example_data$pupil_change <- round(
  sin(row_index / 7),
  4
)
example_data$repetition <- NULL
manifest <- create_gazepoint_feature_manifest(
  features = c("fixation_duration", "pupil_change"),
  scientific_source = c(
```

```

      "Gazepoint fixation export",
      "Gazepoint pupil export"
    ),
    source_table = c("fixations", "pupil"),
    transformation = c(
      "Trial-level mean",
      "Trial-level change"
    ),
    availability_stage = "during_exposure",
    prediction_time_available = TRUE,
    preprocessing_scope = "none",
    fold_local_required = FALSE
  )
  folds <- create_gazepoint_group_folds(
    data = example_data,
    outcome = "outcome",
    predictors = c("fixation_duration", "pupil_change"),
    feature_manifest = manifest,
    generalization_target = "new_participants",
    participant_id = "participant_id",
    trial_id = "trial_id",
    stimulus_id = "stimulus_id",
    v = 3L,
    repeats = 1L,
    seed = 101L
  )
  diagnostics <- diagnose_gazepoint_group_folds(folds)
  print(diagnostics)

```

```
print.gazepoint_fold_diagnostics_validation
```

Print Gazepoint fold-diagnostics validation

Description

Print Gazepoint fold-diagnostics validation

Usage

```
## S3 method for class 'gazepoint_fold_diagnostics_validation'
print(x, ...)
```

Arguments

x	A gazepoint_fold_diagnostics_validation object.
...	Additional arguments, currently unused.

Value

x, invisibly.

Examples

```

example_data <- expand.grid(
  participant_id = sprintf("P%02d", 1:6),
  stimulus_id = sprintf("S%02d", 1:4),
  repetition = 1:2,
  KEEP.OUT.ATTRS = FALSE,
  stringsAsFactors = FALSE
)
example_data$trial_id <- paste0(
  example_data$stimulus_id,
  "_T",
  example_data$repetition
)
participant_number <- as.integer(
  sub("P", "", example_data$participant_id)
)
stimulus_number <- as.integer(
  sub("S", "", example_data$stimulus_id)
)
example_data$outcome <- factor(
  ifelse(
    (participant_number + stimulus_number) %% 2L == 0L,
    "review",
    "pass"
  ),
  levels = c("pass", "review")
)
row_index <- seq_len(nrow(example_data))
example_data$fixation_duration <- 180 + row_index
example_data$pupil_change <- round(
  sin(row_index / 7),
  4
)
example_data$repetition <- NULL
manifest <- create_gazepoint_feature_manifest(
  features = c("fixation_duration", "pupil_change"),
  scientific_source = c(
    "Gazepoint fixation export",
    "Gazepoint pupil export"
  ),
  source_table = c("fixations", "pupil"),
  transformation = c(
    "Trial-level mean",
    "Trial-level change"
  ),
  availability_stage = "during_exposure",
  prediction_time_available = TRUE,
  preprocessing_scope = "none",
  fold_local_required = FALSE
)
folds <- create_gazepoint_group_folds(
  data = example_data,

```

```

outcome = "outcome",
predictors = c("fixation_duration", "pupil_change"),
feature_manifest = manifest,
generalization_target = "new_participants",
participant_id = "participant_id",
trial_id = "trial_id",
stimulus_id = "stimulus_id",
v = 3L,
repeats = 1L,
seed = 101L
)
diagnostics <- diagnose_gazepoint_group_folds(folds)
validation <- validate_gazepoint_fold_diagnostics(
  diagnostics
)
print(validation)

```

```
print.gazepoint_group_folds
```

Print group-aware Gazepoint resampling folds

Description

Print group-aware Gazepoint resampling folds

Usage

```
## S3 method for class 'gazepoint_group_folds'
print(x, ...)
```

Arguments

x	A gazepoint_group_folds object.
...	Additional arguments, currently unused.

Value

x, invisibly.

Examples

```

example_data <- expand.grid(
  participant_id = sprintf("P%02d", 1:6),
  stimulus_id = sprintf("S%02d", 1:4),
  repetition = 1:2,
  KEEP.OUT.ATTRS = FALSE,
  stringsAsFactors = FALSE
)
example_data$trial_id <- paste0(

```

```

    example_data$stimulus_id,
    "_T",
    example_data$repetition
  )
  participant_number <- as.integer(
    sub("P", "", example_data$participant_id)
  )
  stimulus_number <- as.integer(
    sub("S", "", example_data$stimulus_id)
  )
  example_data$outcome <- factor(
    ifelse(
      (participant_number + stimulus_number) %% 2L == 0L,
      "review",
      "pass"
    ),
    levels = c("pass", "review")
  )
  row_index <- seq_len(nrow(example_data))
  example_data$fixation_duration <- 180 + row_index
  example_data$pupil_change <- round(
    sin(row_index / 7),
    4
  )
  )
  example_data$repetition <- NULL
  manifest <- create_gazepoint_feature_manifest(
    features = c("fixation_duration", "pupil_change"),
    scientific_source = c(
      "Gazepoint fixation export",
      "Gazepoint pupil export"
    ),
  ),
  source_table = c("fixations", "pupil"),
  transformation = c(
    "Trial-level mean",
    "Trial-level change"
  ),
  ),
  availability_stage = "during_exposure",
  prediction_time_available = TRUE,
  preprocessing_scope = "none",
  fold_local_required = FALSE
  )
  folds <- create_gazepoint_group_folds(
    data = example_data,
    outcome = "outcome",
    predictors = c("fixation_duration", "pupil_change"),
    feature_manifest = manifest,
    generalization_target = "new_participants",
    participant_id = "participant_id",
    trial_id = "trial_id",
    stimulus_id = "stimulus_id",
    v = 3L,
    repeats = 1L,
    seed = 101L
  )

```

```
)
print(folds)
```

```
print.gazepoint_group_folds_audit
      Print aggregated group-fold leakage auditing
```

Description

Print aggregated group-fold leakage auditing

Usage

```
## S3 method for class 'gazepoint_group_folds_audit'
print(x, ...)
```

Arguments

x	A gazepoint_group_folds_audit object.
...	Additional arguments, currently unused.

Value

x, invisibly.

Examples

```
example_data <- expand.grid(
  participant_id = sprintf("P%02d", 1:6),
  stimulus_id = sprintf("S%02d", 1:4),
  repetition = 1:2,
  KEEP.OUT.ATTRS = FALSE,
  stringsAsFactors = FALSE
)
example_data$trial_id <- paste0(
  example_data$stimulus_id,
  "_T",
  example_data$repetition
)
participant_number <- as.integer(
  sub("P", "", example_data$participant_id)
)
stimulus_number <- as.integer(
  sub("S", "", example_data$stimulus_id)
)
example_data$outcome <- factor(
  ifelse(
    (participant_number + stimulus_number) %% 2L == 0L,
    "review",
```

```

      "pass"
    ),
    levels = c("pass", "review")
  )
  row_index <- seq_len(nrow(example_data))
  example_data$fixation_duration <- 180 + row_index
  example_data$pupil_change <- round(
    sin(row_index / 7),
    4
  )
  example_data$repetition <- NULL
  manifest <- create_gazepoint_feature_manifest(
    features = c("fixation_duration", "pupil_change"),
    scientific_source = c(
      "Gazepoint fixation export",
      "Gazepoint pupil export"
    ),
    source_table = c("fixations", "pupil"),
    transformation = c(
      "Trial-level mean",
      "Trial-level change"
    ),
    availability_stage = "during_exposure",
    prediction_time_available = TRUE,
    preprocessing_scope = "none",
    fold_local_required = FALSE
  )
  folds <- create_gazepoint_group_folds(
    data = example_data,
    outcome = "outcome",
    predictors = c("fixation_duration", "pupil_change"),
    feature_manifest = manifest,
    generalization_target = "new_participants",
    participant_id = "participant_id",
    trial_id = "trial_id",
    stimulus_id = "stimulus_id",
    v = 3L,
    repeats = 1L,
    seed = 101L
  )
  audit <- audit_gazepoint_group_folds(folds)
  print(audit)

```

```
print.gazepoint_group_folds_validation
```

Print group-aware fold validation

Description

Print group-aware fold validation

Usage

```
## S3 method for class 'gazepoint_group_folds_validation'
print(x, ...)
```

Arguments

x A gazepoint_group_folds_validation object.

... Additional arguments, currently unused.

Value

x, invisibly.

Examples

```
example_data <- expand.grid(
  participant_id = sprintf("P%02d", 1:6),
  stimulus_id = sprintf("S%02d", 1:4),
  repetition = 1:2,
  KEEP.OUT.ATTRS = FALSE,
  stringsAsFactors = FALSE
)
example_data$trial_id <- paste0(
  example_data$stimulus_id,
  "_T",
  example_data$repetition
)
participant_number <- as.integer(
  sub("P", "", example_data$participant_id)
)
stimulus_number <- as.integer(
  sub("S", "", example_data$stimulus_id)
)
example_data$outcome <- factor(
  ifelse(
    (participant_number + stimulus_number) %% 2L == 0L,
    "review",
    "pass"
  ),
  levels = c("pass", "review")
)
row_index <- seq_len(nrow(example_data))
example_data$fixation_duration <- 180 + row_index
example_data$pupil_change <- round(
  sin(row_index / 7),
  4
)
example_data$repetition <- NULL
manifest <- create_gazepoint_feature_manifest(
  features = c("fixation_duration", "pupil_change"),
  scientific_source = c(
```

```

    "Gazepoint fixation export",
    "Gazepoint pupil export"
  ),
  source_table = c("fixations", "pupil"),
  transformation = c(
    "Trial-level mean",
    "Trial-level change"
  ),
  availability_stage = "during_exposure",
  prediction_time_available = TRUE,
  preprocessing_scope = "none",
  fold_local_required = FALSE
)
folds <- create_gazepoint_group_folds(
  data = example_data,
  outcome = "outcome",
  predictors = c("fixation_duration", "pupil_change"),
  feature_manifest = manifest,
  generalization_target = "new_participants",
  participant_id = "participant_id",
  trial_id = "trial_id",
  stimulus_id = "stimulus_id",
  v = 3L,
  repeats = 1L,
  seed = 101L
)
validation <- validate_gazepoint_group_folds(folds)
print(validation)

```

```
print.gazepoint_ml_leakage_audit
```

Print a Gazepoint ML leakage audit

Description

Print a Gazepoint ML leakage audit

Usage

```
## S3 method for class 'gazepoint_ml_leakage_audit'
print(x, ...)
```

Arguments

x	An object returned by <code>audit_gazepoint_ml_leakage()</code> .
...	Additional arguments, currently unused.

Value

x, invisibly.

Examples

```

analysis <- data.frame(
  participant_id = c("P01", "P01", "P02", "P02"),
  trial_id = c("T01", "T02", "T03", "T04"),
  stimulus_id = c("S01", "S02", "S03", "S04"),
  outcome = c(0, 1, 0, 1),
  fixation_duration = c(210, 240, 225, 260),
  pupil_change = c(0.10, 0.16, 0.12, 0.18)
)
assessment <- data.frame(
  participant_id = c("P03", "P03", "P04", "P04"),
  trial_id = c("T05", "T06", "T07", "T08"),
  stimulus_id = c("S05", "S06", "S07", "S08"),
  outcome = c(1, 0, 1, 0),
  fixation_duration = c(275, 230, 290, 245),
  pupil_change = c(0.21, 0.11, 0.24, 0.14)
)
audit <- audit_gazepoint_ml_leakage(
  analysis = analysis,
  assessment = assessment,
  outcome = "outcome",
  predictors = c("fixation_duration", "pupil_change"),
  participant_id = "participant_id",
  trial_id = "trial_id",
  stimulus_id = "stimulus_id",
  generalization_target = "new_participants"
)
print(audit)

```

```
print.gazepoint_ml_split
```

Print a group-aware Gazepoint split

Description

Print a group-aware Gazepoint split

Usage

```
## S3 method for class 'gazepoint_ml_split'
print(x, ...)
```

Arguments

x	A gazepoint_ml_split object.
...	Additional arguments, currently unused.

Value

x, invisibly.

Examples

```

example_data <- expand.grid(
  participant_id = sprintf("P%02d", 1:6),
  stimulus_id = sprintf("S%02d", 1:4),
  repetition = 1:2,
  KEEP.OUT.ATTRS = FALSE,
  stringsAsFactors = FALSE
)
example_data$trial_id <- paste0(
  example_data$stimulus_id,
  "_T",
  example_data$repetition
)
participant_number <- as.integer(
  sub("P", "", example_data$participant_id)
)
stimulus_number <- as.integer(
  sub("S", "", example_data$stimulus_id)
)
example_data$outcome <- factor(
  ifelse(
    (participant_number + stimulus_number) %% 2L == 0L,
    "review",
    "pass"
  ),
  levels = c("pass", "review")
)
row_index <- seq_len(nrow(example_data))
example_data$fixation_duration <- 180 + row_index
example_data$pupil_change <- round(
  sin(row_index / 7),
  4
)
example_data$repetition <- NULL
manifest <- create_gazepoint_feature_manifest(
  features = c("fixation_duration", "pupil_change"),
  scientific_source = c(
    "Gazepoint fixation export",
    "Gazepoint pupil export"
  ),
  source_table = c("fixations", "pupil"),
  transformation = c(
    "Trial-level mean",
    "Trial-level change"
  ),
  availability_stage = "during_exposure",
  prediction_time_available = TRUE,
  preprocessing_scope = "none",
  fold_local_required = FALSE
)
split <- split_gazepoint_ml_data(
  data = example_data,

```

```

outcome = "outcome",
predictors = c("fixation_duration", "pupil_change"),
feature_manifest = manifest,
generalization_target = "new_participants",
participant_id = "participant_id",
trial_id = "trial_id",
stimulus_id = "stimulus_id",
assessment_prop = 1 / 3,
seed = 101L
)
print(split)

```

```
print.gazepoint_ml_split_validation
```

Print group-aware split validation

Description

Print group-aware split validation

Usage

```

## S3 method for class 'gazepoint_ml_split_validation'
print(x, ...)

```

Arguments

x	An object returned by <code>validate_gazepoint_ml_split()</code> .
...	Additional arguments, currently unused.

Value

x, invisibly.

Examples

```

example_data <- expand.grid(
  participant_id = sprintf("P%02d", 1:6),
  stimulus_id = sprintf("S%02d", 1:4),
  repetition = 1:2,
  KEEP.OUT.ATTRS = FALSE,
  stringsAsFactors = FALSE
)
example_data$trial_id <- paste0(
  example_data$stimulus_id,
  "_T",
  example_data$repetition
)
participant_number <- as.integer(

```

```

    sub("P", "", example_data$participant_id)
  )
  stimulus_number <- as.integer(
    sub("S", "", example_data$stimulus_id)
  )
  example_data$outcome <- factor(
    ifelse(
      (participant_number + stimulus_number) %% 2L == 0L,
      "review",
      "pass"
    ),
    levels = c("pass", "review")
  )
  row_index <- seq_len(nrow(example_data))
  example_data$fixation_duration <- 180 + row_index
  example_data$pupil_change <- round(
    sin(row_index / 7),
    4
  )
  example_data$repetition <- NULL
  manifest <- create_gazepoint_feature_manifest(
    features = c("fixation_duration", "pupil_change"),
    scientific_source = c(
      "Gazepoint fixation export",
      "Gazepoint pupil export"
    ),
  ),
  source_table = c("fixations", "pupil"),
  transformation = c(
    "Trial-level mean",
    "Trial-level change"
  ),
  availability_stage = "during_exposure",
  prediction_time_available = TRUE,
  preprocessing_scope = "none",
  fold_local_required = FALSE
)
split <- split_gazepoint_ml_data(
  data = example_data,
  outcome = "outcome",
  predictors = c("fixation_duration", "pupil_change"),
  feature_manifest = manifest,
  generalization_target = "new_participants",
  participant_id = "participant_id",
  trial_id = "trial_id",
  stimulus_id = "stimulus_id",
  assessment_prop = 1 / 3,
  seed = 101L
)
validation <- validate_gazepoint_ml_split(split)
print(validation)

```

```
restore_gazepoint_model_artifact
```

Restore a model artifact

Description

Restore a model artifact

Usage

```
restore_gazepoint_model_artifact(artifact)
```

Arguments

artifact A model artifact.

Value

A restored artifact with an unbundled model where needed.

```
select_gazepoint_model
```

Select a governed candidate using an explicit metric and direction

Description

This function records a reviewable decision. It does not refit a model and refuses accuracy as the sole primary metric.

Usage

```
select_gazepoint_model(  
  x,  
  metric,  
  direction,  
  minimum_success_prop = 0.8,  
  tie_breakers = NULL,  
  rationale  
)  
  
## S3 method for class 'gp3ml_model_selection'  
print(x, ...)
```

Arguments

x	A gp3ml_model_tuning object.
metric	Explicit primary metric.
direction	Explicit optimization direction.
minimum_success_prop	Minimum successful-fold proportion.
tie_breakers	Optional ordered secondary metric names.
rationale	Required human-readable selection rationale.
...	Additional arguments passed to the print method.

Value

A gp3ml_model_selection object.

select_gazepoint_threshold

Select a threshold from a governed threshold evaluation

Description

Select a threshold from a governed threshold evaluation

Usage

```
select_gazepoint_threshold(
  evaluation,
  metric,
  direction = c("maximize", "minimize"),
  threshold_origin = c("inner_resampling", "training"),
  training_partition = "inner_resampling",
  generalization_target,
  scientific_justification,
  abstention_allowed = FALSE,
  abstention_interval = NULL
)
```

Arguments

evaluation	A gp3ml_threshold_evaluation.
metric	Metric column to optimize.
direction	"maximize" or "minimize".
threshold_origin	Must identify an analysis/training source.

training_partition
 Partition used to select the threshold.
 generalization_target
 Declared target.
 scientific_justification
 Explicit justification.
 abstention_allowed
 Whether abstention is allowed.
 abstention_interval
 Optional abstention interval.

Value

A gp3ml_decision_rule.

simulate_gazepoint_governed_data

Simulate governed synthetic Gazepoint-derived data

Description

Creates deterministic, non-sensitive synthetic data for package examples, tests, and website articles. The generated outcomes are explicitly observed: a predefined recording-quality review status, an experimentally assigned condition, and a non-sensitive recorded response.

Usage

```
simulate_gazepoint_governed_data(
  n_participants = 30L,
  n_stimuli = 8L,
  trials_per_cell = 2L,
  seed = 1L
)
```

Arguments

n_participants Number of synthetic participants.
 n_stimuli Number of synthetic stimuli.
 trials_per_cell Number of trials per participant-stimulus cell.
 seed Deterministic random seed.

Value

A data frame containing identifiers, observed outcomes, and predeclared synthetic predictors.

Examples

```
synthetic <- simulate_gazeport_governed_data(
  n_participants = 12L,
  n_stimuli = 4L,
  trials_per_cell = 1L,
  seed = 101L
)
table(synthetic$quality_status)
```

```
simulate_gazeport_research_handoffs
```

Simulate realistic cross-package Gazeport research handoffs

Description

Generates deterministic, shareable, synthetic prepared outputs representing handoffs from gp3tools, gpbiometrics, and gp3sequences. The outcome is an experimentally assigned condition. Bio-metrics variables are signal-quality summaries only; no health, emotion, stress, cognition, or other mental-state outcome is generated or inferred.

Usage

```
simulate_gazeport_research_handoffs(
  n_participants = 24L,
  n_stimuli = 6L,
  trials_per_stimulus = 1L,
  seed = 3001L
)
```

Arguments

n_participants	Number of participants.
n_stimuli	Number of stimuli.
trials_per_stimulus	Trials per participant-stimulus cell.
seed	Deterministic seed.

Value

A gp3ml_research_bundle.

split_gazepoint_ml_data

Create a deterministic group-aware Gazepoint holdout split

Description

Creates analysis and assessment partitions that preserve the grouping unit implied by an explicit generalization target.

Usage

```
split_gazepoint_ml_data(
  data,
  outcome,
  predictors,
  feature_manifest,
  generalization_target,
  participant_id = NULL,
  trial_id = NULL,
  stimulus_id = NULL,
  assessment_prop = 0.2,
  seed = 1L,
  source_row_id = ".gp3ml_source_row"
)
```

Arguments

data	Data frame containing the outcome, predictors, and grouping identifiers.
outcome	Name of the outcome column.
predictors	Character vector of predictor-column names.
feature_manifest	Feature manifest containing the predictors.
generalization_target	Declared predictive-generalization target.
participant_id	Optional participant-identifier column.
trial_id	Optional trial-identifier column.
stimulus_id	Optional stimulus-identifier column.
assessment_prop	Requested assessment proportion.
seed	Integer random seed.
source_row_id	Name of the source-row identifier added to the returned partitions.

Details

For simultaneous participant and stimulus generalization, cross-block rows are placed in the excluded partition.

This function does not perform preprocessing, feature selection, resampling, or model fitting.

Value

An object of class `gazepoint_ml_split`.

Examples

```
example_data <- expand.grid(
  participant_id = sprintf("P%02d", 1:6),
  stimulus_id = sprintf("S%02d", 1:4),
  repetition = 1:2,
  KEEP.OUT.ATTRS = FALSE,
  stringsAsFactors = FALSE
)
example_data$trial_id <- paste0(
  example_data$stimulus_id,
  "_T",
  example_data$repetition
)
participant_number <- as.integer(
  sub("P", "", example_data$participant_id)
)
stimulus_number <- as.integer(
  sub("S", "", example_data$stimulus_id)
)
example_data$outcome <- factor(
  ifelse(
    (participant_number + stimulus_number) %% 2L == 0L,
    "review",
    "pass"
  ),
  levels = c("pass", "review")
)
row_index <- seq_len(nrow(example_data))
example_data$fixation_duration <- 180 + row_index
example_data$pupil_change <- round(
  sin(row_index / 7),
  4
)
example_data$repetition <- NULL
manifest <- create_gazepoint_feature_manifest(
  features = c("fixation_duration", "pupil_change"),
  scientific_source = c(
    "Gazepoint fixation export",
    "Gazepoint pupil export"
  ),
  source_table = c("fixations", "pupil"),
```

```

    transformation = c(
      "Trial-level mean",
      "Trial-level change"
    ),
    availability_stage = "during_exposure",
    prediction_time_available = TRUE,
    preprocessing_scope = "none",
    fold_local_required = FALSE
  )
split <- split_gazepoint_ml_data(
  data = example_data,
  outcome = "outcome",
  predictors = c("fixation_duration", "pupil_change"),
  feature_manifest = manifest,
  generalization_target = "new_participants",
  participant_id = "participant_id",
  trial_id = "trial_id",
  stimulus_id = "stimulus_id",
  assessment_prop = 1 / 3,
  seed = 101L
)
split

```

summarize_gazepoint_resample_performance

Summarize repeated grouped-resampling performance

Description

Summarize repeated grouped-resampling performance

Usage

```

summarize_gazepoint_resample_performance(
  x,
  aggregation = c("fold_distribution", "pooled_rows"),
  conf_level = 0.95
)

## S3 method for class 'gp3ml_resample_performance_summary'
print(x, ...)

```

Arguments

<code>x</code>	A <code>gp3ml_resample_evaluation</code> .
<code>aggregation</code>	Either fold-distribution summaries or pooled row-level predictions. Pooled rows are explicitly labelled and do not change the generalization unit.
<code>conf_level</code>	Confidence level for fold-distribution quantiles.
<code>...</code>	Additional arguments passed to the print method.

Value

A gp3ml_resample_performance_summary object.

```
summarize_gazepoint_resample_uncertainty
```

Summarize uncertainty across folds or repeats

Description

Summarize uncertainty across folds or repeats

Usage

```
summarize_gazepoint_resample_uncertainty(
  evaluation,
  unit = c("fold", "repeat"),
  conf_level = 0.95
)

## S3 method for class 'gp3ml_resample_uncertainty'
print(x, ...)
```

Arguments

evaluation	A gp3ml_resample_evaluation or gp3ml_nested_evaluation.
unit	Distribution unit: individual folds or repeat means.
conf_level	Quantile interval level.
x	An object returned by the corresponding gp3ml constructor, evaluator, summarizer, or validator.
...	Additional arguments passed to the print method.

Value

A gp3ml_resample_uncertainty object.

`summarize_gazepoint_shift`*Summarize dataset shift without collapsing it to one drift score*

Description

Summarize dataset shift without collapsing it to one drift score

Usage

```
summarize_gazepoint_shift(shift, missingness = NULL)
```

Arguments

<code>shift</code>	A dataset-shift audit.
<code>missingness</code>	Optional missingness-shift audit.

Value

A structured summary.

`test_gazepoint_model_portability`*Test model-artifact serialization and optional fresh-process prediction*

Description

Test model-artifact serialization and optional fresh-process prediction

Usage

```
test_gazepoint_model_portability(  
  artifact,  
  newdata = artifact$reference_data,  
  tolerance = 1e-08,  
  fresh_process = FALSE  
)
```

Arguments

<code>artifact</code>	Model artifact.
<code>newdata</code>	Optional prediction fixture.
<code>tolerance</code>	Numeric prediction tolerance.
<code>fresh_process</code>	Whether to test in a fresh R process using <code>callr</code> .

Value

A gp3ml_model_portability_test.

train_gazepoint_classifier

Generic governed binary-classifier training wrapper

Description

Generic governed binary-classifier training wrapper

Usage

```
train_gazepoint_classifier(data, task, predictors = NULL, engine = "glm", ...)
```

Arguments

data	Analysis data used to train the classifier.
task	A governed binary-classification task.
predictors	Optional character vector of predictor columns.
engine	Classification engine name or custom engine.
...	Additional arguments passed to fit_gazepoint_model().

Value

A governed classification gp3ml_model object returned by fit_gazepoint_model().

Examples

```
example_data <- data.frame(
  participant_id = rep(sprintf("P%02d", 1:12), each = 2),
  trial_id = sprintf("T%02d", 1:24),
  stimulus_id = rep(c("S01", "S02"), 12),
  condition = rep(c("A", "B"), 12),
  fixation_duration = 180 + seq_len(24),
  pupil_change = sin(seq_len(24) / 3),
  stringsAsFactors = FALSE
)
example_data$quality_status <- factor(
  c(
    "pass", "review", "pass", "review", "review", "pass",
    "review", "pass", "pass", "review", "review", "pass",
    "review", "pass", "review", "pass", "pass", "review",
    "pass", "review", "review", "pass", "pass", "review"
  ),
  levels = c("pass", "review")
)
```

```

task <- declare_gazepoint_task(
  data = example_data,
  outcome = "quality_status",
  purpose = "Predict predefined recording-quality review status",
  task_type = "classification",
  unit_id = "trial_id",
  participant_id = "participant_id",
  stimulus_id = "stimulus_id",
  generalization_target = "new_participants",
  positive = "review"
)
model <- train_gazepoint_classifier(
  data = example_data,
  task = task,
  predictors = c("fixation_duration", "pupil_change"),
  engine = "glm",
  seed = 101L
)
model

```

tune_gazepoint_model *Evaluate every governed candidate on the same grouped folds*

Description

Evaluate every governed candidate on the same grouped folds

Usage

```

tune_gazepoint_model(
  folds,
  task,
  tuning_grid,
  predictors = NULL,
  metrics = NULL,
  seed = 1L,
  continue_on_error = TRUE,
  keep_evaluations = TRUE
)

## S3 method for class 'gp3ml_model_tuning'
print(x, ...)

```

Arguments

folds	A gazepoint_group_folds object.
task	A governed task.
tuning_grid	A gp3ml_tuning_grid.

predictors	Optional declared predictors.
metrics	Optional metric names retained in the comparison table.
seed	Base deterministic seed.
continue_on_error	Whether failed candidates remain in the result while later candidates continue.
keep_evaluations	Whether complete candidate evaluations are retained.
x	An object returned by the corresponding gp3ml constructor, evaluator, summarizer, or validator.
...	Additional arguments passed to the print method.

Value

A gp3ml_model_tuning object retaining all candidates and failures.

Examples

```
data <- simulate_gazepoint_governed_data(12L, 4L, 1L, 202L)
predictors <- c("tracking_ratio", "blink_rate", "gaze_dispersion")
manifest <- create_gazepoint_synthetic_manifest("quality_status", predictors)
folds <- create_gazepoint_group_folds(
  data, "quality_status", predictors, manifest,
  "new_participants", "participant_id", "trial_id", "stimulus_id",
  v = 3L, repeats = 1L, seed = 202L
)
task <- create_gazepoint_synthetic_task(data, "recording_quality", "new_participants")
grid <- create_gazepoint_tuning_grid(
  "glm",
  preprocessor_grid = list(center = c(TRUE, FALSE), scale = TRUE),
  thresholds = 0.5
)
tuned <- tune_gazepoint_model(folds, task, grid, predictors, seed = 202L)
tuned
```

```
validate_gazepoint_analysis_plan
```

Validate an analysis plan

Description

Validate an analysis plan

Usage

```
validate_gazepoint_analysis_plan(plan)
```

Arguments

plan A gp3ml_analysis_plan.

Value

A validation object.

validate_gazepoint_conformal
Validate a conformal fit

Description

Validate a conformal fit

Usage

validate_gazepoint_conformal(object)

Arguments

object A conformal fit.

Value

A validation object.

validate_gazepoint_decision_rule
Validate a governed decision rule

Description

Validate a governed decision rule

Usage

validate_gazepoint_decision_rule(rule, require_threshold = FALSE)

Arguments

rule A gp3ml_decision_rule.
require_threshold Whether a concrete threshold is required.

Value

A validation object.

```
validate_gazepoint_environment
```

Validate the current environment against a reference record

Description

Validate the current environment against a reference record

Usage

```
validate_gazepoint_environment(reference, root = ".", include_renv = FALSE)
```

Arguments

reference	Reference environment record.
root	Project root.
include_renv	Whether to compare renv lock hashes.

Value

An environment comparison.

```
validate_gazepoint_feature_manifest
```

Validate a Gazepoint feature-provenance manifest

Description

Validates the schema and declared scientific safeguards in a feature manifest. Schema errors stop execution. Substantive concerns are returned as structured pass, review, or fail checks.

Usage

```
validate_gazepoint_feature_manifest(x)
```

Arguments

x	A feature manifest created by <code>create_gazepoint_feature_manifest()</code> or a compatible data frame.
---	--

Details

A manifest fails when an intended predictor is declared as outcome-derived, post-outcome, unavailable at prediction time, or an identifier. It also fails when fold-local estimation is required but preprocessing is declared outside the resampling fold.

Unknown or incomplete provenance is returned for review rather than treated as evidence that a safeguard was satisfied.

Value

An object of class `gazepoint_feature_manifest_validation` containing the overall status, complete checks, non-passing issues, and validated manifest.

Examples

```
manifest <- create_gazepoint_feature_manifest(
  features = "fixation_duration",
  scientific_source = "Gazepoint fixation export",
  source_table = "fixations",
  transformation = "Trial-level mean",
  availability_stage = "during_exposure",
  prediction_time_available = TRUE,
  preprocessing_scope = "none",
  fold_local_required = FALSE
)

validate_gazepoint_feature_manifest(manifest)
```

```
validate_gazepoint_fold_diagnostics
```

Validate Gazepoint fold diagnostics

Description

Validate Gazepoint fold diagnostics

Usage

```
validate_gazepoint_fold_diagnostics(x)
```

Arguments

`x` A `gazepoint_fold_diagnostics` object.

Value

An object of class `gazepoint_fold_diagnostics_validation`.

Examples

```
example_data <- expand.grid(
  participant_id = sprintf("P%02d", 1:6),
  stimulus_id = sprintf("S%02d", 1:4),
  repetition = 1:2,
  KEEP.OUT.ATTRS = FALSE,
  stringsAsFactors = FALSE
)
```

```

example_data$trial_id <- paste0(
  example_data$stimulus_id,
  "_T",
  example_data$repetition
)
participant_number <- as.integer(
  sub("P", "", example_data$participant_id)
)
stimulus_number <- as.integer(
  sub("S", "", example_data$stimulus_id)
)
example_data$outcome <- factor(
  ifelse(
    (participant_number + stimulus_number) %% 2L == 0L,
    "review",
    "pass"
  ),
  levels = c("pass", "review")
)
row_index <- seq_len(nrow(example_data))
example_data$fixation_duration <- 180 + row_index
example_data$pupil_change <- round(
  sin(row_index / 7),
  4
)
example_data$repetition <- NULL
manifest <- create_gazepoint_feature_manifest(
  features = c("fixation_duration", "pupil_change"),
  scientific_source = c(
    "Gazepoint fixation export",
    "Gazepoint pupil export"
  ),
  source_table = c("fixations", "pupil"),
  transformation = c(
    "Trial-level mean",
    "Trial-level change"
  ),
  availability_stage = "during_exposure",
  prediction_time_available = TRUE,
  preprocessing_scope = "none",
  fold_local_required = FALSE
)
folds <- create_gazepoint_group_folds(
  data = example_data,
  outcome = "outcome",
  predictors = c("fixation_duration", "pupil_change"),
  feature_manifest = manifest,
  generalization_target = "new_participants",
  participant_id = "participant_id",
  trial_id = "trial_id",
  stimulus_id = "stimulus_id",
  v = 3L,
  repeats = 1L,

```

```

    seed = 101L
  )
  diagnostics <- diagnose_gazepoint_group_folds(folds)
  validation <- validate_gazepoint_fold_diagnostics(
    diagnostics
  )
  validation

```

```
validate_gazepoint_group_folds
```

Validate group-aware Gazepoint resampling folds

Description

Validate group-aware Gazepoint resampling folds

Usage

```
validate_gazepoint_group_folds(x)
```

Arguments

`x` A gazepoint_group_folds object.

Value

An object of class `gazepoint_group_folds_validation`.

Examples

```

example_data <- expand.grid(
  participant_id = sprintf("P%02d", 1:6),
  stimulus_id = sprintf("S%02d", 1:4),
  repetition = 1:2,
  KEEP.OUT.ATTRS = FALSE,
  stringsAsFactors = FALSE
)
example_data$trial_id <- paste0(
  example_data$stimulus_id,
  "_T",
  example_data$repetition
)
participant_number <- as.integer(
  sub("P", "", example_data$participant_id)
)
stimulus_number <- as.integer(
  sub("S", "", example_data$stimulus_id)
)
example_data$outcome <- factor(
  ifelse(

```

```

      (participant_number + stimulus_number) %% 2L == 0L,
      "review",
      "pass"
    ),
    levels = c("pass", "review")
  )
  row_index <- seq_len(nrow(example_data))
  example_data$fixation_duration <- 180 + row_index
  example_data$pupil_change <- round(
    sin(row_index / 7),
    4
  )
  )
  example_data$repetition <- NULL
  manifest <- create_gazepoint_feature_manifest(
    features = c("fixation_duration", "pupil_change"),
    scientific_source = c(
      "Gazepoint fixation export",
      "Gazepoint pupil export"
    ),
    ),
    source_table = c("fixations", "pupil"),
    transformation = c(
      "Trial-level mean",
      "Trial-level change"
    ),
    ),
    availability_stage = "during_exposure",
    prediction_time_available = TRUE,
    preprocessing_scope = "none",
    fold_local_required = FALSE
  )
  folds <- create_gazepoint_group_folds(
    data = example_data,
    outcome = "outcome",
    predictors = c("fixation_duration", "pupil_change"),
    feature_manifest = manifest,
    generalization_target = "new_participants",
    participant_id = "participant_id",
    trial_id = "trial_id",
    stimulus_id = "stimulus_id",
    v = 3L,
    repeats = 1L,
    seed = 101L
  )
  validation <- validate_gazepoint_group_folds(folds)
  validation

```

Description

Validate a Gazepoint handoff

Usage

```
validate_gazepoint_handoff(x)
```

Arguments

x A gp3ml_handoff.

Value

A gp3ml_handoff_validation object.

validate_gazepoint_ml_roles

Validate outcome, predictor, identifier, and grouping roles

Description

Validate outcome, predictor, identifier, and grouping roles

Usage

```
validate_gazepoint_ml_roles(data, task, predictors, feature_manifest = NULL)
```

Arguments

data A data frame containing outcome, predictors, and identifiers.

task A governed gp3ml_task object.

predictors Character vector naming intended predictors.

feature_manifest Optional Gazepoint feature-provenance manifest.

Value

A gp3ml_role_validation object containing the overall status, complete check table, non-passing issues, and optional feature-manifest validation.

Examples

```

example_data <- data.frame(
  participant_id = rep(sprintf("P%02d", 1:12), each = 2),
  trial_id = sprintf("T%02d", 1:24),
  stimulus_id = rep(c("S01", "S02"), 12),
  condition = rep(c("A", "B"), 12),
  fixation_duration = 180 + seq_len(24),
  pupil_change = sin(seq_len(24) / 3),
  stringsAsFactors = FALSE
)
example_data$quality_status <- factor(
  c(
    "pass", "review", "pass", "review", "review", "pass",
    "review", "pass", "pass", "review", "review", "pass",
    "review", "pass", "review", "pass", "pass", "review",
    "pass", "review", "review", "pass", "pass", "review"
  ),
  levels = c("pass", "review")
)
task <- declare_gazepoint_task(
  data = example_data,
  outcome = "quality_status",
  purpose = "Predict predefined recording-quality review status",
  task_type = "classification",
  unit_id = "trial_id",
  participant_id = "participant_id",
  stimulus_id = "stimulus_id",
  generalization_target = "new_participants",
  positive = "review"
)
manifest <- create_gazepoint_feature_manifest(
  features = c("fixation_duration", "pupil_change"),
  scientific_source = c(
    "Gazepoint fixation export",
    "Gazepoint all-gaze export"
  ),
  source_table = c("fixations", "all_gaze"),
  transformation = c(
    "Trial-level mean",
    "Baseline-adjusted change"
  ),
  availability_stage = "during_exposure",
  prediction_time_available = TRUE,
  preprocessing_scope = c("none", "resampling_fold"),
  fold_local_required = c(FALSE, TRUE)
)

validate_gazepoint_ml_roles(
  data = example_data,
  task = task,
  predictors = c("fixation_duration", "pupil_change"),
  feature_manifest = manifest
)

```

)

validate_gazepoint_ml_split

Validate a group-aware Gazepoint holdout split

Description

Validate a group-aware Gazepoint holdout split

Usage

```
validate_gazepoint_ml_split(x)
```

Arguments

x An object returned by `split_gazepoint_ml_data()`.

Value

An object of class `gazepoint_ml_split_validation`.

Examples

```
example_data <- expand.grid(
  participant_id = sprintf("P%02d", 1:6),
  stimulus_id = sprintf("S%02d", 1:4),
  repetition = 1:2,
  KEEP.OUT.ATTRS = FALSE,
  stringsAsFactors = FALSE
)
example_data$trial_id <- paste0(
  example_data$stimulus_id,
  "_T",
  example_data$repetition
)
participant_number <- as.integer(
  sub("P", "", example_data$participant_id)
)
stimulus_number <- as.integer(
  sub("S", "", example_data$stimulus_id)
)
example_data$outcome <- factor(
  ifelse(
    (participant_number + stimulus_number) %% 2L == 0L,
    "review",
    "pass"
  ),
  levels = c("pass", "review")
)
```

```

row_index <- seq_len(nrow(example_data))
example_data$fixation_duration <- 180 + row_index
example_data$pupil_change <- round(
  sin(row_index / 7),
  4
)
example_data$repetition <- NULL
manifest <- create_gazepoint_feature_manifest(
  features = c("fixation_duration", "pupil_change"),
  scientific_source = c(
    "Gazepoint fixation export",
    "Gazepoint pupil export"
  ),
  source_table = c("fixations", "pupil"),
  transformation = c(
    "Trial-level mean",
    "Trial-level change"
  ),
  availability_stage = "during_exposure",
  prediction_time_available = TRUE,
  preprocessing_scope = "none",
  fold_local_required = FALSE
)
split <- split_gazepoint_ml_data(
  data = example_data,
  outcome = "outcome",
  predictors = c("fixation_duration", "pupil_change"),
  feature_manifest = manifest,
  generalization_target = "new_participants",
  participant_id = "participant_id",
  trial_id = "trial_id",
  stimulus_id = "stimulus_id",
  assessment_prop = 1 / 3,
  seed = 101L
)
validation <- validate_gazepoint_ml_split(split)
validation

```

```
validate_gazepoint_model_artifact
```

Validate a model artifact

Description

Validate a model artifact

Usage

```
validate_gazepoint_model_artifact(artifact, verify_hash = TRUE)
```

Arguments

artifact	Artifact to validate.
verify_hash	Whether to recompute the SHA-256 payload hash.

Value

A validation object.

validate_gazepoint_model_tuning
<i>Validate governed tuning results</i>

Description

Validate governed tuning results

Usage

```
validate_gazepoint_model_tuning(x)

## S3 method for class 'gp3ml_model_tuning_validation'
print(x, ...)
```

Arguments

x	A gp3ml_model_tuning object.
...	Additional arguments passed to the print method.

Value

A gp3ml_model_tuning_validation.

validate_gazepoint_nested_evaluation
<i>Validate a nested evaluation</i>

Description

Validate a nested evaluation

Usage

```
validate_gazepoint_nested_evaluation(x)

## S3 method for class 'gp3ml_nested_evaluation_validation'
print(x, ...)
```

Arguments

- x A gp3ml_nested_evaluation.
- ... Additional arguments passed to the print method.

Value

A gp3ml_nested_evaluation_validation.

validate_gazepoint_nested_folds
<i>Validate nested grouped folds</i>

Description

Validate nested grouped folds

Usage

```
validate_gazepoint_nested_folds(x)

## S3 method for class 'gp3ml_nested_folds_validation'
print(x, ...)
```

Arguments

- x A gp3ml_nested_folds object.
- ... Additional arguments passed to the print method.

Value

A gp3ml_nested_folds_validation object.

validate_gazepoint_release_checksums
<i>Validate SHA-256 release checksums</i>

Description

Validate SHA-256 release checksums

Usage

```
validate_gazepoint_release_checksums(manifest, directory = ".")
```

Arguments

manifest	Checksum manifest or path.
directory	Directory containing artifacts.

Value

A validation object.

validate_gazepoint_resample_evaluation
<i>Validate a grouped-fold evaluation result</i>

Description

Validate a grouped-fold evaluation result

Usage

```
validate_gazepoint_resample_evaluation(x)

## S3 method for class 'gp3ml_resample_evaluation_validation'
print(x, ...)
```

Arguments

x	A gp3ml_resample_evaluation.
...	Additional arguments passed to the print method.

Value

A gp3ml_resample_evaluation_validation object.

validate_gazepoint_research_bundle
<i>Validate a synthetic cross-package research bundle</i>

Description

Validate a synthetic cross-package research bundle

Usage

```
validate_gazepoint_research_bundle(x)
```

Arguments

x A gp3ml_research_bundle.

Value

A gp3ml_research_bundle_validation.

validate_gazepoint_ro_crate

Validate a gp3ml RO-Crate-oriented export

Description

Validate a gp3ml RO-Crate-oriented export

Usage

validate_gazepoint_ro_crate(path)

Arguments

path Crate directory or gp3ml_ro_crate.

Value

A validation object.

validate_gazepoint_target_uncertainty

Validate target-aligned uncertainty metadata

Description

Validate target-aligned uncertainty metadata

Usage

validate_gazepoint_target_uncertainty(x)

```
## S3 method for class 'gp3ml_uncertainty_validation'
print(x, ...)
```

Arguments

x A gp3ml_target_uncertainty or gp3ml_resample_uncertainty.
 ... Additional arguments passed to the print method.

Value

A gp3ml_uncertainty_validation.

```
validate_gazepoint_transportability
```

Validate an external transportability report

Description

Validate an external transportability report

Usage

```
validate_gazepoint_transportability(x)

## S3 method for class 'gp3ml_transportability_validation'
print(x, ...)
```

Arguments

x	A gp3ml_transportability_report.
...	Additional arguments passed to the print method.

Value

A gp3ml_transportability_validation.

```
validate_gp3ml_object_contract
```

Validate an object against the gp3ml public-object contract

Description

Validate an object against the gp3ml public-object contract

Usage

```
validate_gp3ml_object_contract(x, registry = gp3ml_api_contracts())
```

Arguments

x	A gp3ml object.
registry	Contract registry from gp3ml_api_contracts().

Value

A gp3ml_object_contract_validation object.

`with_gazepoint_reproducible_output`*Evaluate code with deterministic documentation-output settings*

Description

Evaluate code with deterministic documentation-output settings

Usage

```
with_gazepoint_reproducible_output(code)
```

Arguments

<code>code</code>	Expression to evaluate.
-------------------	-------------------------

Value

The value of code.

`write_external_validation_report`*Write an external-validation report*

Description

Write an external-validation report

Usage

```
write_external_validation_report(report, path, overwrite = FALSE)
```

Arguments

<code>report</code>	A <code>gp3ml_external_validation_report</code> object.
<code>path</code>	Destination Markdown file path.
<code>overwrite</code>	Whether an existing file may be replaced.

Value

The destination path, returned invisibly after the Markdown report is written.

Examples

```

training_data <- data.frame(
  participant_id = rep(sprintf("P%02d", 1:12), each = 2),
  trial_id = sprintf("T%02d", 1:24),
  stimulus_id = rep(c("S01", "S02"), 12),
  fixation_duration = 180 + seq_len(24),
  pupil_change = sin(seq_len(24) / 3),
  stringsAsFactors = FALSE
)
training_data$quality_status <- factor(
  c(
    "pass", "review", "pass", "review", "review", "pass",
    "review", "pass", "pass", "review", "review", "pass",
    "review", "pass", "review", "pass", "pass", "review",
    "pass", "review", "review", "pass", "pass", "review"
  ),
  levels = c("pass", "review")
)
task <- declare_gazepoint_task(
  data = training_data,
  outcome = "quality_status",
  purpose = "Predict predefined recording-quality review status",
  task_type = "classification",
  unit_id = "trial_id",
  participant_id = "participant_id",
  stimulus_id = "stimulus_id",
  generalization_target = "new_participants",
  positive = "review"
)
model <- train_gazepoint_classifier(
  data = training_data,
  task = task,
  predictors = c("fixation_duration", "pupil_change"),
  engine = "glm",
  seed = 101L
)
external_data <- training_data
external_data$participant_id <- rep(
  sprintf("E%02d", 1:12),
  each = 2
)
external_data$trial_id <- sprintf("ET%02d", 1:24)
external_data$fixation_duration <-
  external_data$fixation_duration + 4
external_data$pupil_change <- cos(seq_len(24) / 4)
validation <- evaluate_external_validation(
  model = model,
  external_data = external_data,
  label = "synthetic_external",
  bootstrap = 10L,
  seed = 101L
)

```

```
report <- create_external_validation_report(validation)
output <- tempfile(fileext = ".md")
write_external_validation_report(report, output)
file.exists(output)
unlink(output)
```

```
write_gazepoint_analysis_plan
```

Write an analysis plan

Description

Write an analysis plan

Usage

```
write_gazepoint_analysis_plan(plan, path, format = c("rds", "json", "md"))
```

Arguments

plan	Analysis plan.
path	Output path.
format	"rds", "json", or "md".

Value

Normalized output path, invisibly.

```
write_gazepoint_feature_manifest_csv
```

Write a Gazepoint feature manifest or validation table to CSV

Description

Writes a feature manifest or one table from a validated manifest to a UTF-8 CSV file. Existing files are not replaced unless explicitly permitted.

Usage

```
write_gazepoint_feature_manifest_csv(
  x,
  file,
  table = c("manifest", "issues", "checks"),
  overwrite = FALSE,
  na = ""
)
```

Arguments

x	A gazepoint_feature_manifest, compatible data frame, or object returned by <code>validate_gazepoint_feature_manifest()</code> .
file	A single output path ending in .csv.
table	Table to export. One of "manifest", "issues", or "checks". Plain manifest inputs support only "manifest".
overwrite	Logical. When FALSE, the default, an existing file causes an error.
na	Character value used for missing values.

Value

The normalized output path, invisibly.

Examples

```
manifest <- create_gazepoint_feature_manifest(
  features = "fixation_duration",
  scientific_source = "Gazepoint fixation export",
  source_table = "fixations",
  transformation = "Trial-level mean",
  availability_stage = "during_exposure",
  prediction_time_available = TRUE,
  preprocessing_scope = "none",
  fold_local_required = FALSE
)

output <- tempfile(fileext = ".csv")

write_gazepoint_feature_manifest_csv(
  manifest,
  output
)

unlink(output)
```

```
write_gazepoint_fold_diagnostics_csv
```

Write Gazepoint fold diagnostics to CSV files

Description

Write Gazepoint fold diagnostics to CSV files

Usage

```
write_gazepoint_fold_diagnostics_csv(
  x,
  directory,
  prefix = "gazepoint_fold_diagnostics",
  tables = c("fold_metrics", "repeat_metrics", "outcome_balance", "group_balance",
    "assessment_coverage", "exclusion_summary", "validation_checks", "validation_issues"),
  overwrite = FALSE,
  na = ""
)
```

Arguments

x	A gazepoint_fold_diagnostics object.
directory	Output directory.
prefix	File-name prefix.
tables	Diagnostic tables to export.
overwrite	Whether existing files may be overwritten.
na	String used for missing values.

Value

A named character vector of written file paths, invisibly.

Examples

```
example_data <- expand.grid(
  participant_id = sprintf("P%02d", 1:6),
  stimulus_id = sprintf("S%02d", 1:4),
  repetition = 1:2,
  KEEP.OUT.ATTRS = FALSE,
  stringsAsFactors = FALSE
)
example_data$trial_id <- paste0(
  example_data$stimulus_id,
  "_T",
  example_data$repetition
)
participant_number <- as.integer(
  sub("P", "", example_data$participant_id)
)
stimulus_number <- as.integer(
  sub("S", "", example_data$stimulus_id)
)
example_data$outcome <- factor(
  ifelse(
    (participant_number + stimulus_number) %% 2L == 0L,
    "review",
    "pass"
  )
)
```

```

    ),
    levels = c("pass", "review")
  )
  row_index <- seq_len(nrow(example_data))
  example_data$fixation_duration <- 180 + row_index
  example_data$pupil_change <- round(
    sin(row_index / 7),
    4
  )
  )
  example_data$repetition <- NULL
  manifest <- create_gazepoint_feature_manifest(
    features = c("fixation_duration", "pupil_change"),
    scientific_source = c(
      "Gazepoint fixation export",
      "Gazepoint pupil export"
    ),
  ),
  source_table = c("fixations", "pupil"),
  transformation = c(
    "Trial-level mean",
    "Trial-level change"
  ),
  ),
  availability_stage = "during_exposure",
  prediction_time_available = TRUE,
  preprocessing_scope = "none",
  fold_local_required = FALSE
)
folds <- create_gazepoint_group_folds(
  data = example_data,
  outcome = "outcome",
  predictors = c("fixation_duration", "pupil_change"),
  feature_manifest = manifest,
  generalization_target = "new_participants",
  participant_id = "participant_id",
  trial_id = "trial_id",
  stimulus_id = "stimulus_id",
  v = 3L,
  repeats = 1L,
  seed = 101L
)
diagnostics <- diagnose_gazepoint_group_folds(folds)
output_directory <- tempfile()
paths <- write_gazepoint_fold_diagnostics_csv(
  x = diagnostics,
  directory = output_directory,
  tables = c("fold_metrics", "repeat_metrics")
)
basename(unname(paths))
unlink(output_directory, recursive = TRUE)

```

```
write_gazepoint_group_folds_csv
```

Write group-aware resampling tables to CSV

Description

Write group-aware resampling tables to CSV

Usage

```
write_gazepoint_group_folds_csv(
  x,
  directory,
  prefix = "gazepoint_group_folds",
  tables = c("assignments", "fold_summary", "group_counts", "group_mapping",
    "validation_checks", "validation_issues", "audit_summary", "audit_checks",
    "audit_issues"),
  include_fold_data = FALSE,
  overwrite = FALSE,
  na = ""
)
```

Arguments

<code>x</code>	A <code>gazepoint_group_folds</code> object.
<code>directory</code>	Output directory.
<code>prefix</code>	Non-empty filename prefix.
<code>tables</code>	Character vector selecting summary tables.
<code>include_fold_data</code>	Logical. Whether every materialized fold partition should also be written.
<code>overwrite</code>	Logical. Whether existing files may be replaced.
<code>na</code>	Character representation of missing values.

Value

A named character vector of normalized output paths, invisibly.

Examples

```
example_data <- expand.grid(
  participant_id = sprintf("P%02d", 1:6),
  stimulus_id = sprintf("S%02d", 1:4),
  repetition = 1:2,
  KEEP.OUT.ATTRS = FALSE,
  stringsAsFactors = FALSE
)
example_data$trial_id <- paste0(
  example_data$stimulus_id,
```

```

      "_T",
      example_data$repetition
    )
    participant_number <- as.integer(
      sub("P", "", example_data$participant_id)
    )
    stimulus_number <- as.integer(
      sub("S", "", example_data$stimulus_id)
    )
    example_data$outcome <- factor(
      ifelse(
        (participant_number + stimulus_number) %% 2L == 0L,
        "review",
        "pass"
      ),
      levels = c("pass", "review")
    )
    row_index <- seq_len(nrow(example_data))
    example_data$fixation_duration <- 180 + row_index
    example_data$pupil_change <- round(
      sin(row_index / 7),
      4
    )
  )
  example_data$repetition <- NULL
  manifest <- create_gazepoint_feature_manifest(
    features = c("fixation_duration", "pupil_change"),
    scientific_source = c(
      "Gazepoint fixation export",
      "Gazepoint pupil export"
    ),
    source_table = c("fixations", "pupil"),
    transformation = c(
      "Trial-level mean",
      "Trial-level change"
    ),
    availability_stage = "during_exposure",
    prediction_time_available = TRUE,
    preprocessing_scope = "none",
    fold_local_required = FALSE
  )
  folds <- create_gazepoint_group_folds(
    data = example_data,
    outcome = "outcome",
    predictors = c("fixation_duration", "pupil_change"),
    feature_manifest = manifest,
    generalization_target = "new_participants",
    participant_id = "participant_id",
    trial_id = "trial_id",
    stimulus_id = "stimulus_id",
    v = 3L,
    repeats = 1L,
    seed = 101L
  )

```

```

output_directory <- tempfile()
paths <- write_gazepoint_group_folds_csv(
  x = folds,
  directory = output_directory,
  tables = c("fold_summary", "group_counts")
)
basename(unname(paths))
unlink(output_directory, recursive = TRUE)

```

```
write_gazepoint_ml_leakage_audit_csv
```

Write a Gazepoint ML leakage-audit table to CSV

Description

Writes one machine-readable table from a leakage-audit object to a UTF-8 CSV file. Existing files are not replaced unless explicitly permitted.

Usage

```

write_gazepoint_ml_leakage_audit_csv(
  x,
  file,
  table = c("issues", "checks", "partition_summary"),
  overwrite = FALSE,
  na = ""
)

```

Arguments

<code>x</code>	An object returned by <code>audit_gazepoint_ml_leakage()</code> .
<code>file</code>	A single output file path ending in <code>.csv</code> .
<code>table</code>	The audit table to export. One of "issues", "checks", or "partition_summary".
<code>overwrite</code>	Logical. When FALSE, the default, an existing output file causes an error.
<code>na</code>	Character value used for missing values in the CSV file.

Value

The normalized output path, invisibly.

Examples

```

analysis <- data.frame(
  participant_id = c("P01", "P02"),
  trial_id = c("T01", "T02"),
  outcome = c(0, 1),
  feature = c(1.2, 1.8)
)

```

```

)

assessment <- data.frame(
  participant_id = c("P03", "P04"),
  trial_id = c("T03", "T04"),
  outcome = c(1, 0),
  feature = c(2.1, 2.4)
)

audit <- audit_gazepoint_ml_leakage(
  analysis = analysis,
  assessment = assessment,
  outcome = "outcome",
  predictors = "feature",
  participant_id = "participant_id",
  trial_id = "trial_id",
  generalization_target = "new_participants"
)

output <- tempfile(fileext = ".csv")

write_gazepoint_ml_leakage_audit_csv(
  audit,
  output,
  table = "checks"
)

unlink(output)

```

write_gazepoint_ml_split_csv

Write group-aware split tables to CSV

Description

Write group-aware split tables to CSV

Usage

```

write_gazepoint_ml_split_csv(
  x,
  directory,
  prefix = "gazepoint_ml_split",
  tables = c("analysis", "assessment", "excluded", "assignment", "summary",
    "group_counts", "checks", "issues"),
  overwrite = FALSE,
  na = ""
)

```

Arguments

<code>x</code>	A gazepoint_ml_split object.
<code>directory</code>	Output directory.
<code>prefix</code>	Filename prefix.
<code>tables</code>	Tables to export.
<code>overwrite</code>	Whether existing files may be replaced.
<code>na</code>	Character representation of missing values.

Value

A named character vector of normalized file paths, invisibly.

Examples

```
example_data <- expand.grid(
  participant_id = sprintf("P%02d", 1:6),
  stimulus_id = sprintf("S%02d", 1:4),
  repetition = 1:2,
  KEEP.OUT.ATTRS = FALSE,
  stringsAsFactors = FALSE
)
example_data$trial_id <- paste0(
  example_data$stimulus_id,
  "_T",
  example_data$repetition
)
participant_number <- as.integer(
  sub("P", "", example_data$participant_id)
)
stimulus_number <- as.integer(
  sub("S", "", example_data$stimulus_id)
)
example_data$outcome <- factor(
  ifelse(
    (participant_number + stimulus_number) %% 2L == 0L,
    "review",
    "pass"
  ),
  levels = c("pass", "review")
)
row_index <- seq_len(nrow(example_data))
example_data$fixation_duration <- 180 + row_index
example_data$pupil_change <- round(
  sin(row_index / 7),
  4
)
example_data$repetition <- NULL
manifest <- create_gazepoint_feature_manifest(
  features = c("fixation_duration", "pupil_change"),
  scientific_source = c(
```

```

      "Gazepoint fixation export",
      "Gazepoint pupil export"
    ),
    source_table = c("fixations", "pupil"),
    transformation = c(
      "Trial-level mean",
      "Trial-level change"
    ),
    availability_stage = "during_exposure",
    prediction_time_available = TRUE,
    preprocessing_scope = "none",
    fold_local_required = FALSE
  )
split <- split_gazepoint_ml_data(
  data = example_data,
  outcome = "outcome",
  predictors = c("fixation_duration", "pupil_change"),
  feature_manifest = manifest,
  generalization_target = "new_participants",
  participant_id = "participant_id",
  trial_id = "trial_id",
  stimulus_id = "stimulus_id",
  assessment_prop = 1 / 3,
  seed = 101L
)
output_directory <- tempfile()
paths <- write_gazepoint_ml_split_csv(
  x = split,
  directory = output_directory,
  tables = c("summary", "group_counts")
)
basename(unname(paths))
unlink(output_directory, recursive = TRUE)

```

```
write_gazepoint_model_card
```

Write a model card

Description

Write a model card

Usage

```

write_gazepoint_model_card(
  card,
  path,
  format = c("markdown", "json"),
  overwrite = FALSE
)

```

Arguments

<code>card</code>	A <code>gp3ml_model_card</code> object.
<code>path</code>	Destination file path.
<code>format</code>	Output format: Markdown or JSON.
<code>overwrite</code>	Whether an existing file may be replaced.

Value

The destination path, returned invisibly after the model card is written.

Examples

```
training_data <- data.frame(
  participant_id = rep(sprintf("P%02d", 1:12), each = 2),
  trial_id = sprintf("T%02d", 1:24),
  stimulus_id = rep(c("S01", "S02"), 12),
  fixation_duration = 180 + seq_len(24),
  pupil_change = sin(seq_len(24) / 3),
  stringsAsFactors = FALSE
)
training_data$quality_status <- factor(
  c(
    "pass", "review", "pass", "review", "review", "pass",
    "review", "pass", "pass", "review", "review", "pass",
    "review", "pass", "review", "pass", "pass", "review",
    "pass", "review", "review", "pass", "pass", "review"
  ),
  levels = c("pass", "review")
)
task <- declare_gazepoint_task(
  data = training_data,
  outcome = "quality_status",
  purpose = "Predict predefined recording-quality review status",
  task_type = "classification",
  unit_id = "trial_id",
  participant_id = "participant_id",
  stimulus_id = "stimulus_id",
  generalization_target = "new_participants",
  positive = "review"
)
model <- train_gazepoint_classifier(
  data = training_data,
  task = task,
  predictors = c("fixation_duration", "pupil_change"),
  engine = "glm",
  seed = 101L
)
card <- create_gazepoint_model_card(
  model = model,
  intended_use = paste(
    "Support manual review of predefined",
```

```
      "recording-quality status"
    ),
    limitations = "Synthetic example for documentation."
  )
output <- tempfile(fileext = ".md")
write_gazepoint_model_card(
  card = card,
  path = output,
  format = "markdown"
)
file.exists(output)
unlink(output)
```

write_gazepoint_model_tuning

Write governed tuning and selection tables

Description

Write governed tuning and selection tables

Usage

```
write_gazepoint_model_tuning(
  x,
  directory,
  prefix = "gazepoint_model_tuning",
  selection = NULL,
  overwrite = FALSE
)
```

Arguments

x	A gp3ml_model_tuning object.
directory	Output directory.
prefix	Filename prefix.
selection	Optional gp3ml_model_selection to record.
overwrite	Whether existing files may be replaced.

Value

Named output paths, invisibly.

`write_gazepoint_nested_evaluation`*Write nested-resampling evaluation tables*

Description

Write nested-resampling evaluation tables

Usage

```
write_gazepoint_nested_evaluation(  
    x,  
    directory,  
    prefix = "gazepoint_nested_evaluation",  
    overwrite = FALSE  
)
```

Arguments

<code>x</code>	A <code>gp3ml_nested_evaluation</code> .
<code>directory</code>	Output directory.
<code>prefix</code>	Filename prefix.
<code>overwrite</code>	Whether existing files may be replaced.

Value

Named paths, invisibly.

`write_gazepoint_release_checksums`*Write SHA-256 checksums for release artifacts*

Description

Write SHA-256 checksums for release artifacts

Usage

```
write_gazepoint_release_checksums(files, path = "SHA256SUMS.csv")
```

Arguments

<code>files</code>	Release artifact files.
<code>path</code>	Output checksum manifest.

Value

A gp3ml_release_checksum_manifest.

write_gazepoint_release_model_card

Write a release-ready governed model card

Description

Write a release-ready governed model card

Usage

```
write_gazepoint_release_model_card(  
    card,  
    path,  
    format = c("markdown", "json"),  
    overwrite = FALSE  
)
```

Arguments

card	A gp3ml_release_model_card.
path	Destination path.
format	Markdown or JSON.
overwrite	Whether an existing file may be replaced.

Value

The destination path, invisibly.

write_gazepoint_reproducibility_audit

Write a reproducibility-hardening audit

Description

Write a reproducibility-hardening audit

Usage

```
write_gazepoint_reproducibility_audit(  
    audit,  
    directory = ".",  
    prefix = "gp3ml_reproducibility_audit",  
    overwrite = FALSE  
)
```

Arguments

audit	A gp3ml_reproducibility_audit.
directory	Destination directory.
prefix	File prefix.
overwrite	Whether existing files may be replaced.

Value

Named paths.

```
write_gazepoint_reproducibility_report
```

Write a reproducibility report

Description

Write a reproducibility report

Usage

```
write_gazepoint_reproducibility_report(report, path, overwrite = FALSE)
```

Arguments

report	A gp3ml_reproducibility_report object.
path	Destination Markdown file path.
overwrite	Whether an existing file may be replaced.

Value

The destination path, returned invisibly after the reproducibility report is written.

Examples

```
example_data <- data.frame(
  trial_id = sprintf("T%02d", 1:6),
  fixation_duration = c(190, 205, 198, 214, 202, 220),
  stringsAsFactors = FALSE
)
report <- create_gazepoint_reproducibility_report(
  objects = list(
    fixation_values = example_data$fixation_duration
  ),
  data = example_data,
  seeds = list(example = 101L),
  notes = "Synthetic documentation example.",
  project_path = tempdir()
```

```
)  
output <- tempfile(fileext = ".md")  
write_gazepoint_reproducibility_report(report, output)  
file.exists(output)  
unlink(output)
```

write_gazepoint_resample_evaluation

Write grouped-fold evaluation tables

Description

Write grouped-fold evaluation tables

Usage

```
write_gazepoint_resample_evaluation(  
  x,  
  directory,  
  prefix = "gazepoint_resample_evaluation",  
  overwrite = FALSE  
)
```

Arguments

x	A gp3ml_resample_evaluation.
directory	Output directory.
prefix	Filename prefix.
overwrite	Whether existing files may be replaced.

Value

Named output paths, invisibly.

write_gazepoint_ro_crate

Write a minimal RO-Crate-oriented research object

Description

This helper writes a conservative RO-Crate-oriented JSON-LD metadata file and SHA-256 file hashes. It does not claim formal RO-Crate conformance; use an independent validator when formal conformance is required.

Usage

```
write_gazepoint_ro_crate(
  path,
  files,
  name,
  description,
  creator_name,
  creator_orcid = NULL,
  license = "MIT",
  doi = NULL,
  copy_files = TRUE
)
```

Arguments

path	Output directory.
files	Named or unnamed character vector of files to include.
name	Research-object name.
description	Description.
creator_name	Creator name.
creator_orcid	Optional ORCID URI or identifier.
license	License URI or label.
doi	Optional DOI.
copy_files	Whether to copy files into the crate directory.

Value

A gp3ml_ro_crate.

```
write_gazepoint_target_uncertainty
```

Write target-aligned uncertainty tables

Description

Write target-aligned uncertainty tables

Usage

```
write_gazepoint_target_uncertainty(
  x,
  directory,
  prefix = "gazepoint_target_uncertainty",
  overwrite = FALSE
)
```

Arguments

x	A gp3ml uncertainty object.
directory	Output directory.
prefix	Filename prefix.
overwrite	Whether existing files may be replaced.

Value

Named paths, invisibly.

write_gazepoint_transportability_report
<i>Write an expanded transportability report</i>

Description

Write an expanded transportability report

Usage

```
write_gazepoint_transportability_report(report, path, overwrite = FALSE)
```

Arguments

report	A gp3ml_transportability_report.
path	Destination Markdown path.
overwrite	Whether an existing file may be replaced.

Value

The destination path, invisibly.

```
write_gp3ml_api_contracts
```

Write gp3ml API contracts

Description

Write gp3ml API contracts

Usage

```
write_gp3ml_api_contracts(  
  registry = gp3ml_api_contracts(),  
  directory = ".",  
  prefix = "gp3ml_api_contracts",  
  overwrite = FALSE  
)
```

Arguments

registry	Contract registry.
directory	Destination directory.
prefix	File prefix.
overwrite	Whether existing files may be replaced.

Value

Named character vector of written paths.

```
write_gp3ml_governance_profile
```

Write a governance profile audit

Description

Write a governance profile audit

Usage

```
write_gp3ml_governance_profile(audit, path)
```

Arguments

audit	Governance profile audit.
path	Markdown output path.

write_gp3ml_governance_profile

149

Value

Output path, invisibly.

Index

`apply_gazepoint_calibrator`, 5
`apply_gazepoint_decision_rule`, 6
`as_gp3ml_data`, 7
`assert_gp3ml_engine_available`, 7
`assert_gp3ml_use_case`, 8
`assess_gazepoint_calibration`, 9
`assess_gazepoint_conformal_coverage`, 10
`audit_gazepoint_abstention`, 11
`audit_gazepoint_dataset_shift`, 11
`audit_gazepoint_group_folds`, 12
`audit_gazepoint_missingness_shift`, 13
`audit_gazepoint_ml_leakage`, 14
`audit_gazepoint_ml_leakage()`, 96, 136
`audit_gazepoint_model_robustness`, 16
`audit_gazepoint_nested_resampling`, 17
`audit_gazepoint_plan_deviations`, 17
`audit_gazepoint_reproducibility`, 18
`audit_gp3ml_api_stability`, 19
`audit_gp3ml_governance_profile`, 19

`bake_gazepoint_preprocessor`, 20
`bootstrap_gazepoint_metrics`, 21
`bootstrap_gazepoint_metrics_by_unit`, 22

`capture_gazepoint_environment`, 24
`collect_gazepoint_fold_predictions`, 25
`combine_gazepoint_handoffs`, 25
`compare_gazepoint_environments`, 26
`compare_gazepoint_models`, 26
`create_external_validation_report`, 27
`create_gazepoint_decision_rule`, 28
`create_gazepoint_feature_manifest`, 30
`create_gazepoint_feature_manifest()`, 42, 114
`create_gazepoint_group_folds`, 31
`create_gazepoint_handoff`, 34
`create_gazepoint_model_artifact`, 35
`create_gazepoint_model_card`, 36

`create_gazepoint_nested_folds`, 37
`create_gazepoint_release_evidence`, 39
`create_gazepoint_release_model_card`, 40
`create_gazepoint_reproducibility_report`, 41
`create_gazepoint_synthetic_manifest`, 42
`create_gazepoint_synthetic_task`, 43
`create_gazepoint_tuning_grid`, 44
`create_gp3ml_governance_profile`, 45

`declare_gazepoint_analysis_plan`, 45
`declare_gazepoint_external_dataset`, 47
`declare_gazepoint_task`, 48
`diagnose_gazepoint_group_folds`, 50

`evaluate_external_validation`, 52
`evaluate_gazepoint_external_transportability`, 54
`evaluate_gazepoint_feature_stability`, 55
`evaluate_gazepoint_group_folds`, 55
`evaluate_gazepoint_missingness_sensitivity`, 57
`evaluate_gazepoint_nested_resampling`, 58
`evaluate_gazepoint_seed_stability`, 59
`evaluate_gazepoint_threshold_stability`, 59
`evaluate_gazepoint_thresholds`, 60

`fit_gazepoint_calibrator`, 61
`fit_gazepoint_conformal`, 62
`fit_gazepoint_deep_model`, 63
`fit_gazepoint_model`, 65
`fit_gazepoint_model()`, 56
`fit_gazepoint_preprocessor`, 66
`fit_gazepoint_preprocessor()`, 56

`gazepoint_classification_metrics`, 68

gazeppoint_performance_metrics, 69
gazeppoint_regression_metrics, 70
gp3ml_api_contracts, 71
gp3ml_available_engines, 71
gp3ml_engine_capabilities, 72
gp3ml_interop_contracts, 72
gp3ml_object_schema, 73
gp3ml_prohibited_uses, 73

integrate_black_box_model, 74

lock_gazeppoint_analysis_plan, 75

normalize_gazeppoint_artifact_text, 76

plot.gp3ml_abstention_audit, 76
plot.gp3ml_api_stability_audit, 77
plot.gp3ml_conformal_coverage, 77
plot.gp3ml_dataset_shift_audit, 78
plot.gp3ml_engine_capabilities, 78
plot.gp3ml_environment_comparison, 79
plot.gp3ml_governance_profile_audit, 79
plot.gp3ml_handoff_validation, 80
plot.gp3ml_model_artifact_validation, 80
plot.gp3ml_model_robustness_audit, 81
plot.gp3ml_plan_deviation_audit, 81
plot.gp3ml_release_checksum_validation, 82
plot.gp3ml_reproducibility_audit, 82
plot.gp3ml_research_bundle_validation, 83
plot.gp3ml_ro_crate_validation, 83
plot.gp3ml_threshold_evaluation, 84
predict.gp3ml_model, 84
predict_gazeppoint_interval, 86
predict_gazeppoint_set, 86
print.gazeppoint_feature_manifest_validation, 87
print.gazeppoint_fold_diagnostics, 87
print.gazeppoint_fold_diagnostics_validation, 89
print.gazeppoint_group_folds, 91
print.gazeppoint_group_folds_audit, 93
print.gazeppoint_group_folds_validation, 94
print.gazeppoint_ml_leakage_audit, 96
print.gazeppoint_ml_split, 97

print.gazeppoint_ml_split_validation, 99
print.gp3ml_external_dataset_declaration
(declare_gazeppoint_external_dataset), 47
print.gp3ml_model_selection
(select_gazeppoint_model), 101
print.gp3ml_model_tuning
(tune_gazeppoint_model), 111
print.gp3ml_model_tuning_validation
(validate_gazeppoint_model_tuning), 123
print.gp3ml_nested_evaluation
(evaluate_gazeppoint_nested_resampling), 58
print.gp3ml_nested_evaluation_validation
(validate_gazeppoint_nested_evaluation), 123
print.gp3ml_nested_folds
(create_gazeppoint_nested_folds), 37
print.gp3ml_nested_folds_validation
(validate_gazeppoint_nested_folds), 124
print.gp3ml_nested_resampling_audit
(audit_gazeppoint_nested_resampling), 17
print.gp3ml_release_evidence
(create_gazeppoint_release_evidence), 39
print.gp3ml_release_model_card
(create_gazeppoint_release_model_card), 40
print.gp3ml_resample_evaluation
(evaluate_gazeppoint_group_folds), 55
print.gp3ml_resample_evaluation_validation
(validate_gazeppoint_resample_evaluation), 125
print.gp3ml_resample_performance_summary
(summarize_gazeppoint_resample_performance), 107
print.gp3ml_resample_uncertainty
(summarize_gazeppoint_resample_uncertainty), 108
print.gp3ml_target_uncertainty
(bootstrap_gazeppoint_metrics_by_unit), 23

print_gp3ml_transportability_report
 (evaluate_gazepoint_external_transportability), 54
 print_gp3ml_transportability_validation
 (validate_gazepoint_transportability), 127
 print_gp3ml_tuning_grid
 (create_gazepoint_tuning_grid), 44
 print_gp3ml_uncertainty_validation
 (validate_gazepoint_target_uncertainty), 126
 restore_gazepoint_model_artifact, 101
 select_gazepoint_model, 101
 select_gazepoint_threshold, 102
 simulate_gazepoint_governed_data, 103
 simulate_gazepoint_governed_data(), 43
 simulate_gazepoint_research_handoffs, 104
 split_gazepoint_ml_data, 105
 split_gazepoint_ml_data(), 121
 summarize_gazepoint_resample_performance, 107
 summarize_gazepoint_resample_uncertainty, 108
 summarize_gazepoint_shift, 109
 test_gazepoint_model_portability, 109
 train_gazepoint_classifier, 110
 tune_gazepoint_model, 111
 validate_gazepoint_analysis_plan, 112
 validate_gazepoint_conformal, 113
 validate_gazepoint_decision_rule, 113
 validate_gazepoint_environment, 114
 validate_gazepoint_feature_manifest, 114
 validate_gazepoint_feature_manifest(), 31, 87, 131
 validate_gazepoint_fold_diagnostics, 115
 validate_gazepoint_group_folds, 117
 validate_gazepoint_handoff, 118
 validate_gazepoint_ml_roles, 119
 validate_gazepoint_ml_split, 121
 validate_gazepoint_ml_split(), 99
 validate_gazepoint_model_artifact, 122
 validate_gazepoint_model_tuning, 123
 validate_gazepoint_nested_evaluation, 123
 validate_gazepoint_nested_folds, 124
 validate_gazepoint_release_checksums, 124
 validate_gazepoint_resample_evaluation, 125
 validate_gazepoint_research_bundle, 125
 validate_gazepoint_ro_crate, 126
 validate_gazepoint_target_uncertainty, 126
 validate_gazepoint_transportability, 127
 validate_gp3ml_object_contract, 127
 with_gazepoint_reproducible_output, 128
 write_external_validation_report, 128
 write_gazepoint_analysis_plan, 130
 write_gazepoint_feature_manifest_csv, 130
 write_gazepoint_fold_diagnostics_csv, 131
 write_gazepoint_group_folds_csv, 133
 write_gazepoint_ml_leakage_audit_csv, 136
 write_gazepoint_ml_split_csv, 137
 write_gazepoint_model_card, 139
 write_gazepoint_model_tuning, 141
 write_gazepoint_nested_evaluation, 142
 write_gazepoint_release_checksums, 142
 write_gazepoint_release_model_card, 143
 write_gazepoint_reproducibility_audit, 143
 write_gazepoint_reproducibility_report, 144
 write_gazepoint_resample_evaluation, 145
 write_gazepoint_ro_crate, 145
 write_gazepoint_target_uncertainty, 146
 write_gazepoint_transportability_report, 147
 write_gp3ml_api_contracts, 148
 write_gp3ml_governance_profile, 148