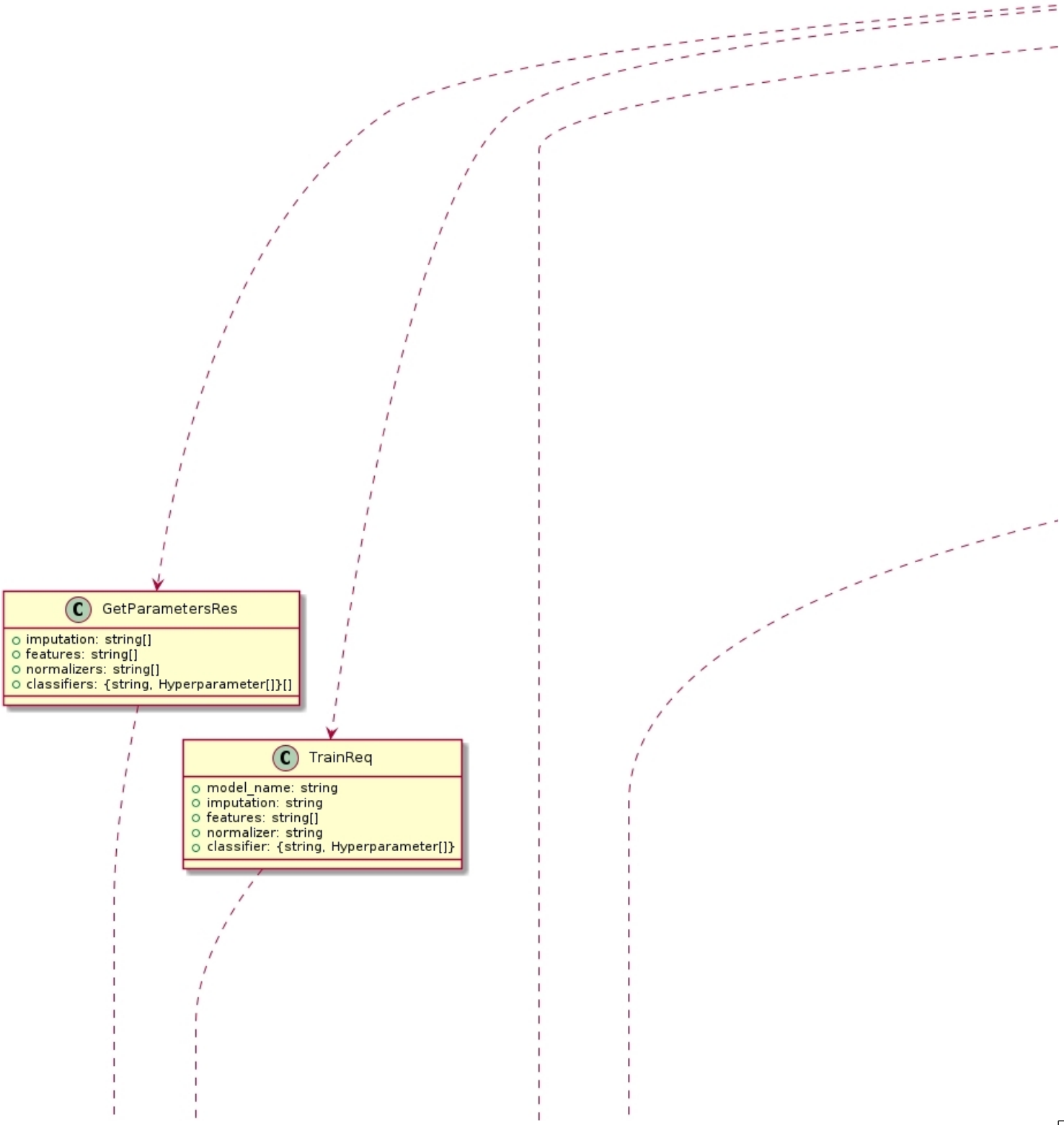
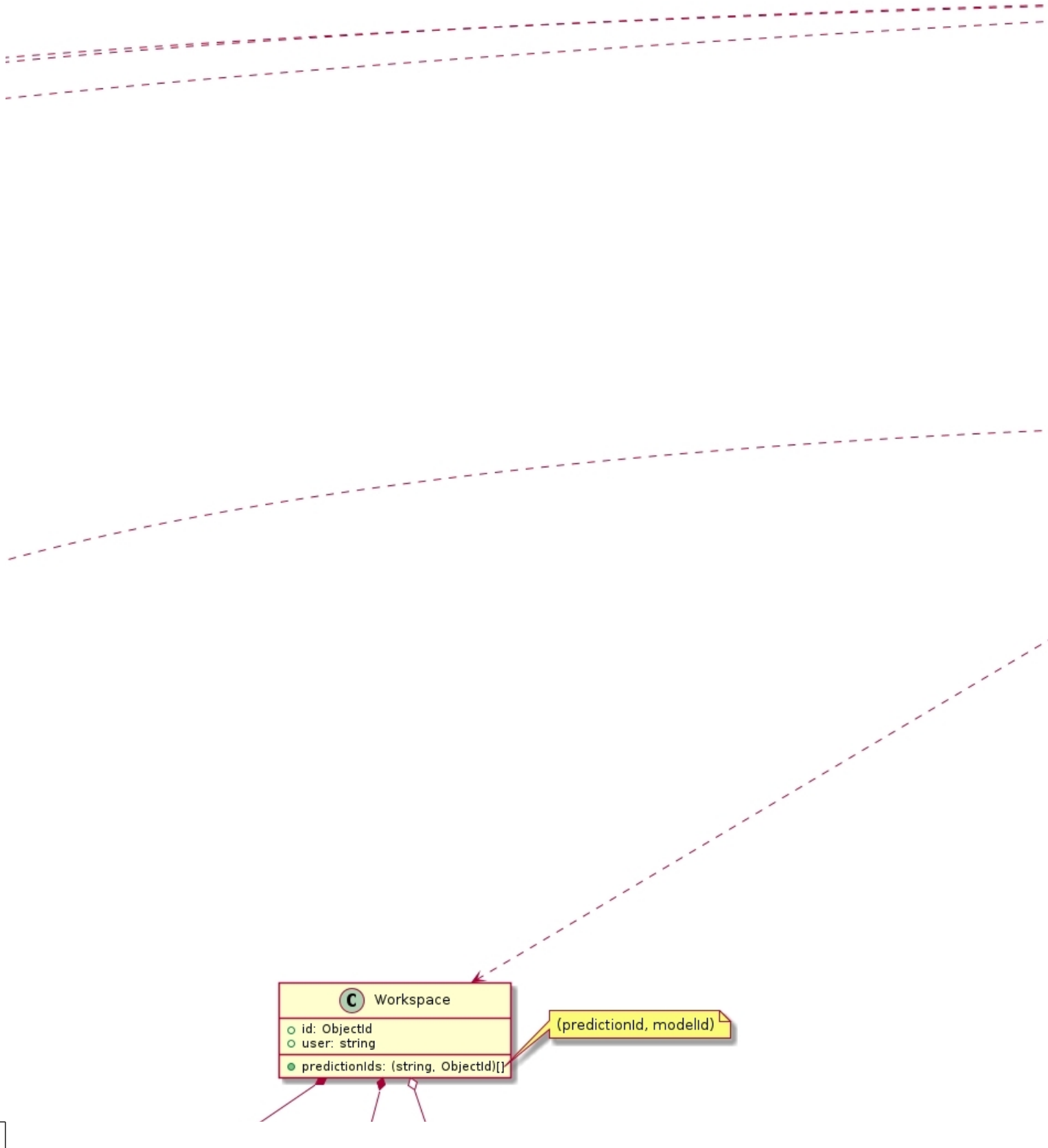
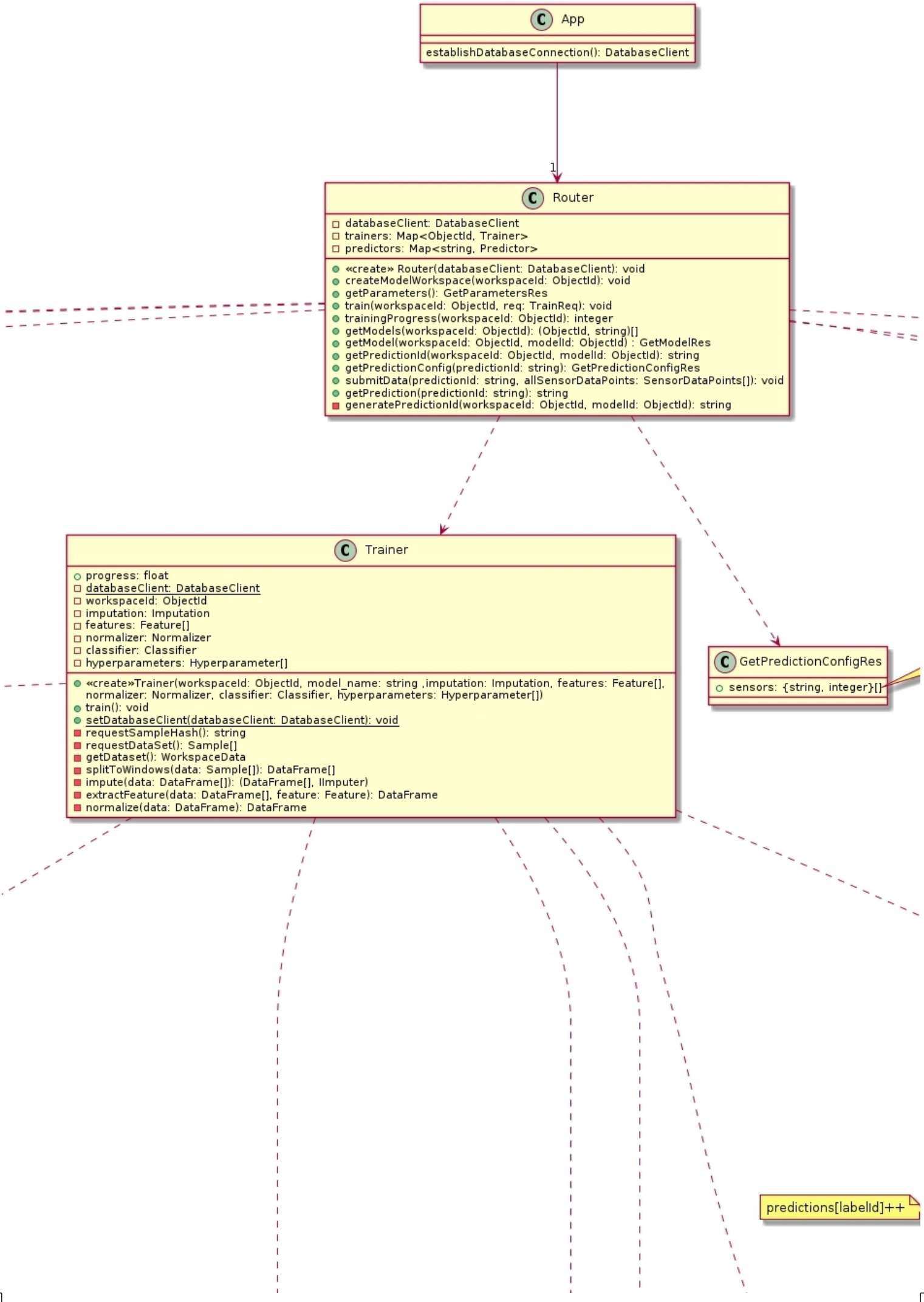
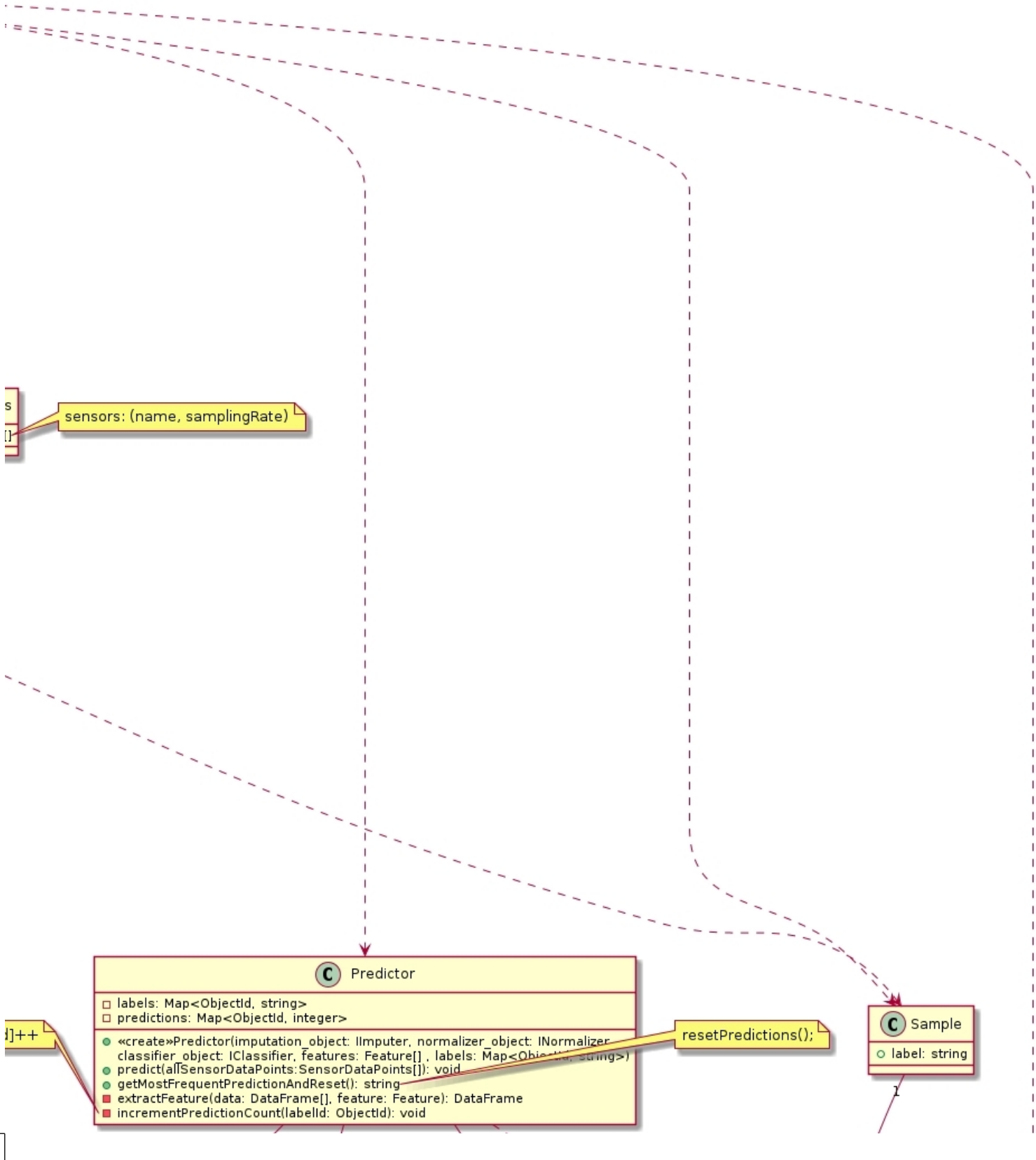
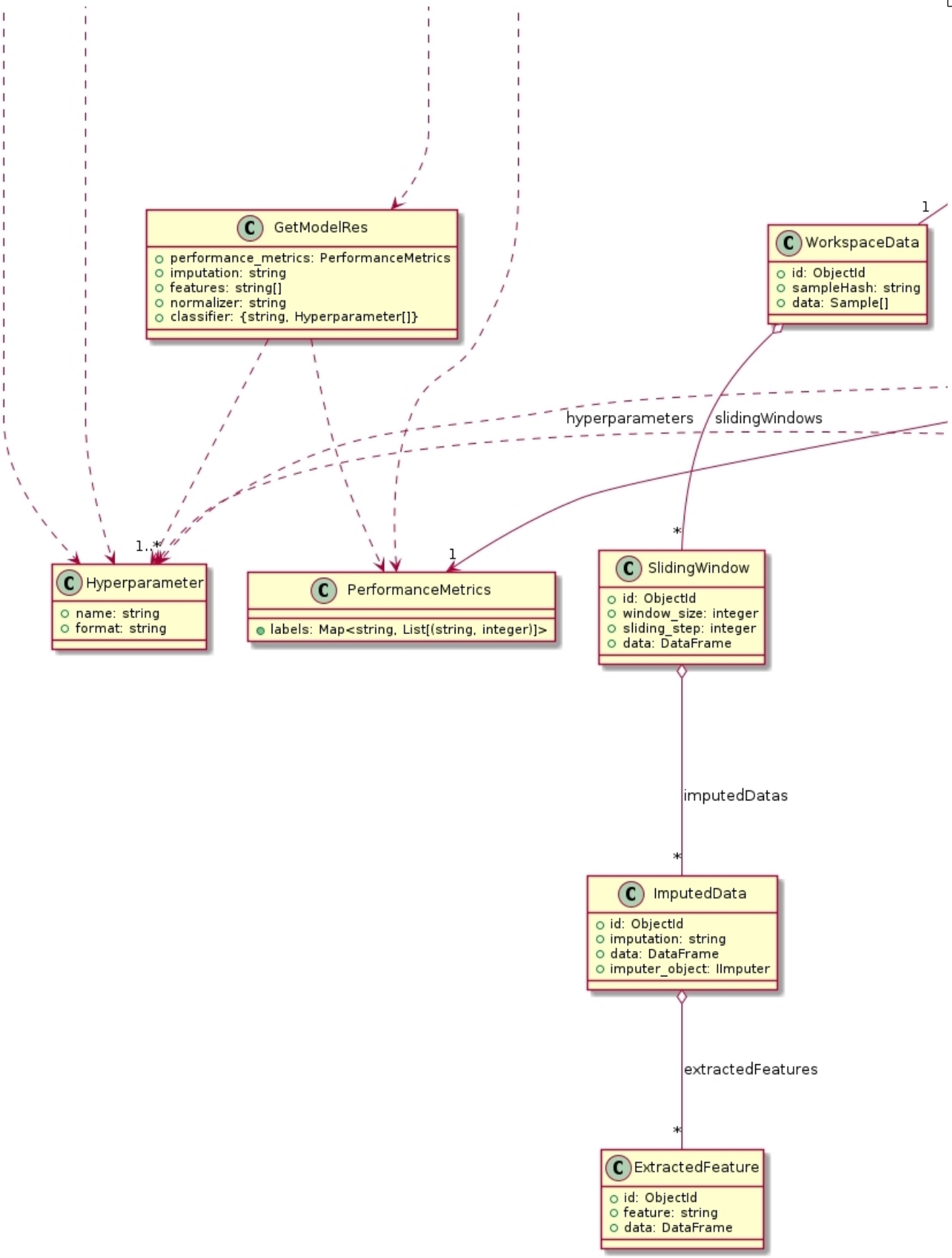
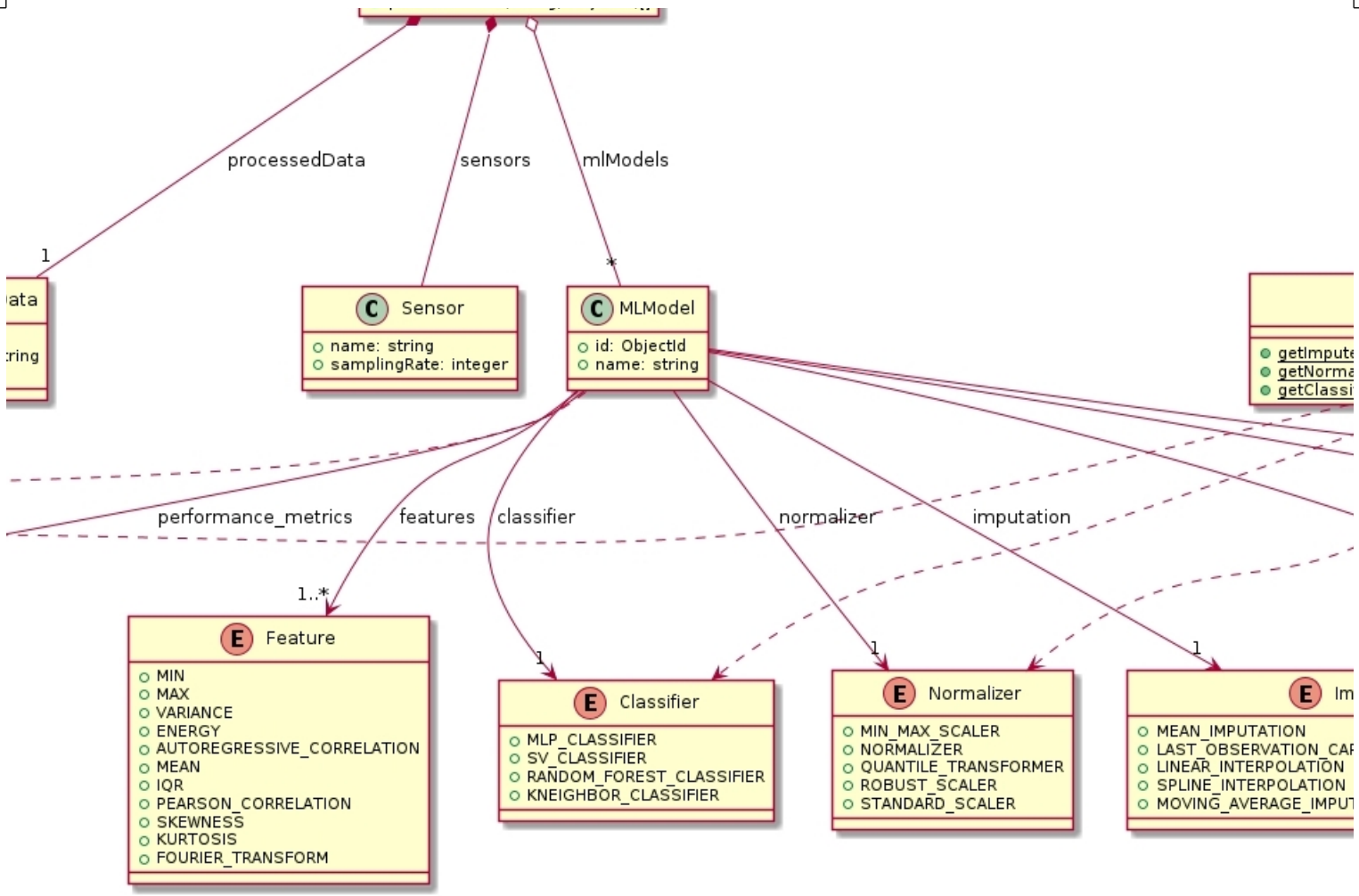

GetParametersRes
imputation: string[]
features: string[]
normalizers: string[]
classifiers: {string, Hyperparameter[]}[]
TrainReq
model_name: string
imputation: string
features: string[]
normalizer: string
classifier: {string, Hyperparameter[]}

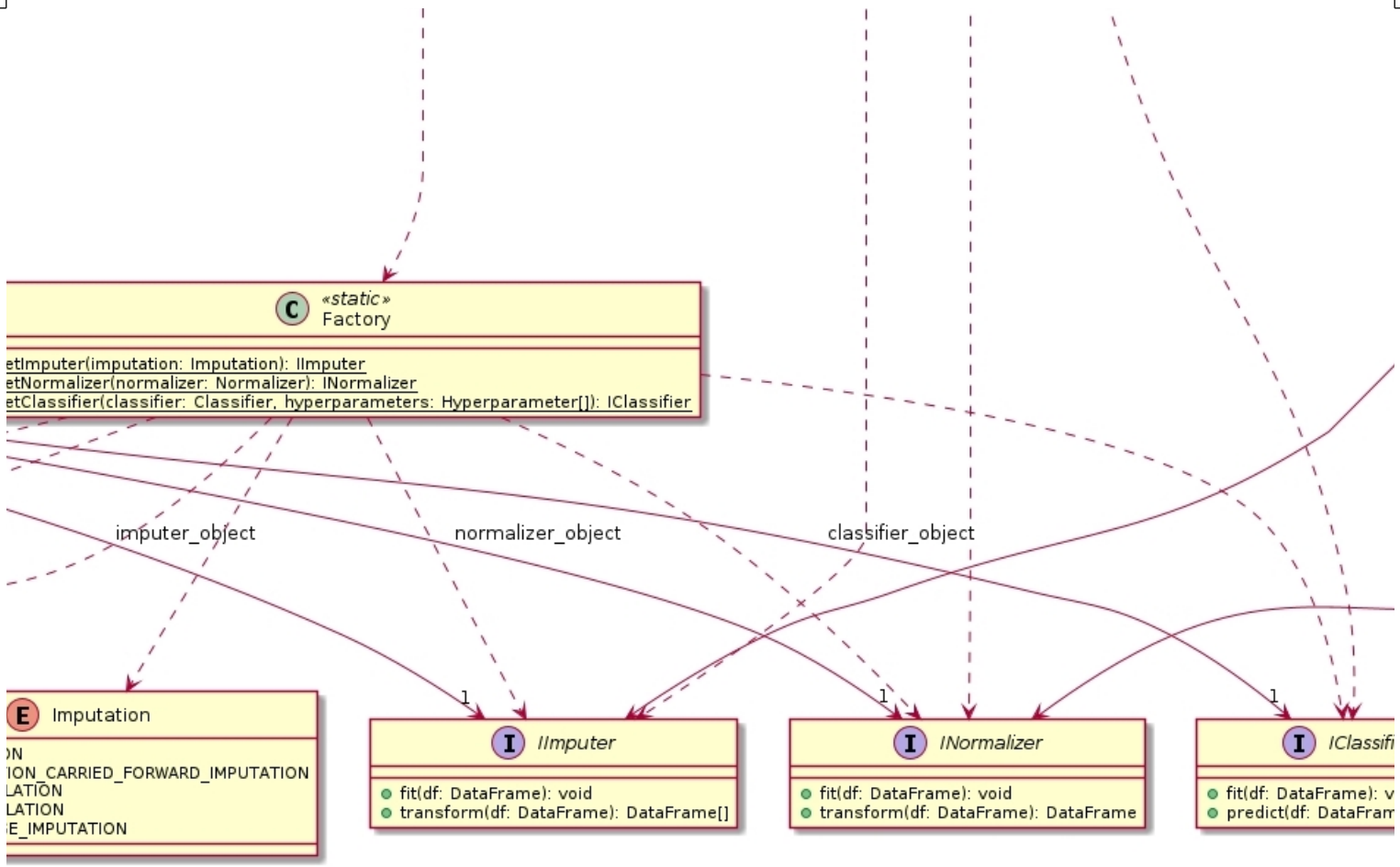

Workspace
id: ObjectId
user: string
predictionIds: (string, ObjectId)[]
(predictionId, modelId)

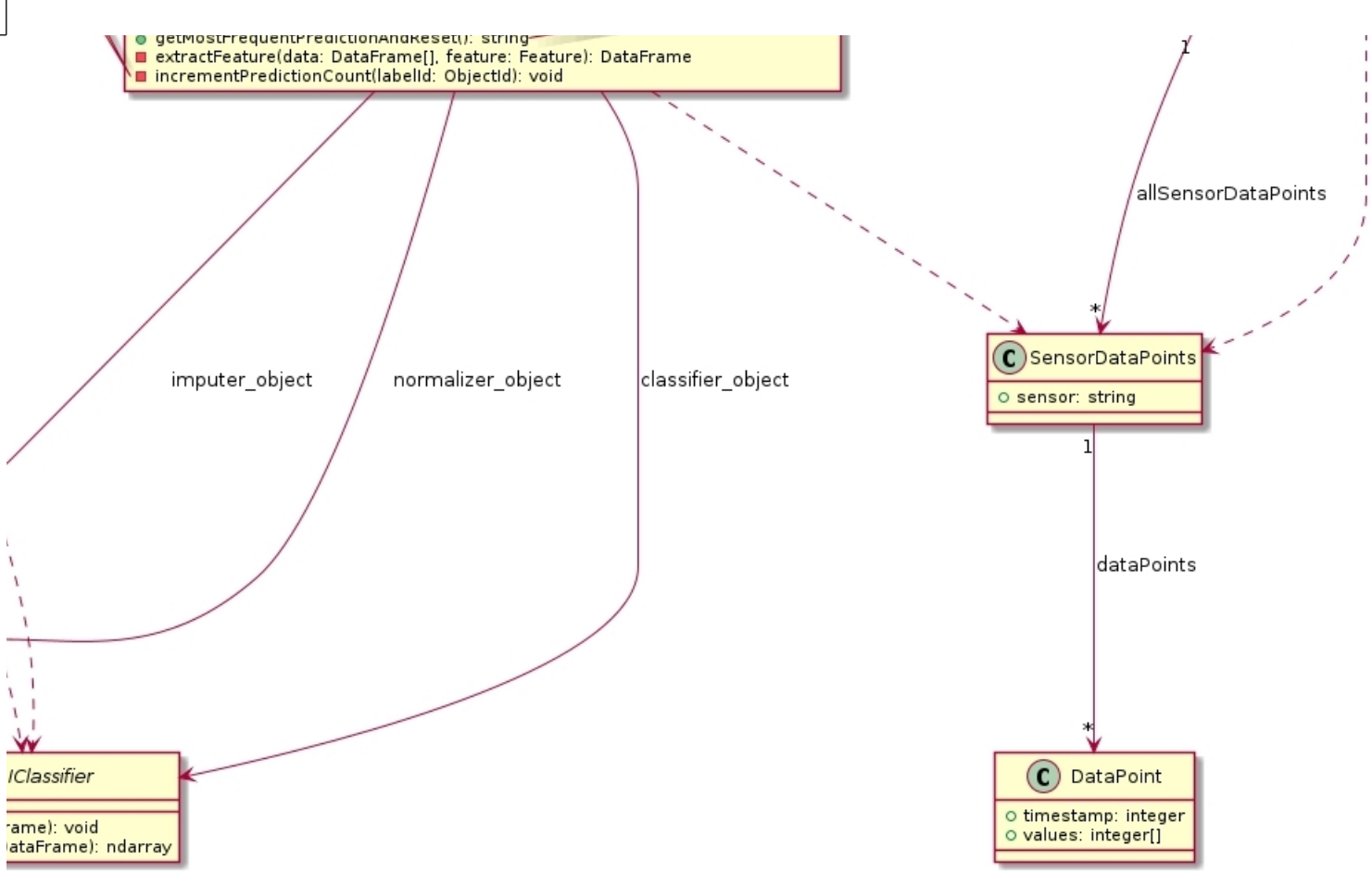

App
establishDatabaseConnection(): DatabaseClient
1
Router
databaseClient: DatabaseClient
trainers: Map<ObjectId, Trainer>
predictors: Map<string, Predictor>
«create» Router(databaseClient: DatabaseClient): void
createModelWorkspace(workspaceId: ObjectId): void
getParameters(): GetParametersRes
train(workspaceId: ObjectId, req: TrainReq): void
trainingProgress(workspaceId: ObjectId): integer
getModels(workspaceId: ObjectId): (ObjectId, string)[]
getModel(workspaceId: ObjectId, modelId: ObjectId) : GetModelRes
getPredictionId(workspaceId: ObjectId, modelId: ObjectId): string
getPredictionConfig(predictionId: string): GetPredictionConfigRes
submitData(predictionId: string, allSensorDataPoints: SensorDataPoints[]): void
getPrediction(predictionId: string): string
generatePredictionId(workspaceId: ObjectId, modelId: ObjectId): string
Trainer
progress: float
databaseClient: DatabaseClient
workspaceId: ObjectId
imputation: Imputation
features: Feature[]
normalizer: Normalizer
classifier: Classifier
hyperparameters: Hyperparameter[]
«create»Trainer(workspaceId: ObjectId, model_name: string ,imputation: Imputation, features: Feature[], normalizer: Normalizer, classifier: Classifier, hyperparameters: Hyperparameter[])
train(): void
setDatabaseClient(databaseClient: DatabaseClient): void
requestSampleHash(): string
requestDataSet(): Sample[]
getDataset(): WorkspaceData
splitToWindows(data: Sample[]): DataFrame[]
impute(data: DataFrame[]): (DataFrame[], IImputer)
extractFeature(data: DataFrame[], feature: Feature): DataFrame
normalize(data: DataFrame): DataFrame
GetPredictionConfigRes
sensors: {string, integer}[]
predictions[labelId]++


sensors: (name, samplingRate)
Predictor
labels: Map<ObjectId, string>
predictions: Map<ObjectId, integer>
«create»Predictor(imputation_object: IImputer, normalizer_object: INormalizer,
classifier_object: IClassifier, features: Feature[] , labels: Map<ObjectId, string>)
predict(allSensorDataPoints:SensorDataPoints[]): void
getMostFrequentPredictionAndReset(): string
extractFeature(data: DataFrame[], feature: Feature): DataFrame
incrementPredictionCount(labelId: ObjectId): void
resetPredictions();
Sample
label: string
1


GetModelRes
performance_metrics: PerformanceMetrics
imputation: string
features: string[]
normalizer: string
classifier: {string, Hyperparameter[]}
WorkspaceData
id: ObjectId
sampleHash: string
data: Sample[]
1
hyperparameters
slidingWindows
1..*
1
*
Hyperparameter
name: string
format: string
PerformanceMetrics
labels: Map<string, List[(string, integer)]>
SlidingWindow
id: ObjectId
window_size: integer
sliding_step: integer
data: DataFrame
imputedDatas
*
ImputedData
id: ObjectId
imputation: string
data: DataFrame
imputer_object: IImputer
extractedFeatures
*
ExtractedFeature
id: ObjectId
feature: string
data: DataFrame


processedData
sensors
mlModels
1
*
Sensor
name: string
samplingRate: integer
MLModel
id: ObjectId
name: string
performance_metrics
features
classifier
normalizer
imputation
1..*
Feature
MIN
MAX
VARIANCE
ENERGY
AUTOREGRESSIVE_CORRELATION
MEAN
IQR
PEARSON_CORRELATION
SKEWNESS
KURTOSIS
FOURIER_TRANSFORM
Classifier
MLP_CLASSIFIER
SV_CLASSIFIER
RANDOM_FOREST_CLASSIFIER
KNEIGHBOR_CLASSIFIER
Normalizer
MIN_MAX_SCALER
NORMALIZER
QUANTILE_TRANSFORMER
ROBUST_SCALER
STANDARD_SCALER
MEAN_IMPUTATION
SPLINE_INTERPOLATION


«static»
Factory
etImputer(imputation: Imputation): IImputer
etNormalizer(normalizer: Normalizer): INormalizer
etClassifier(classifier: Classifier, hyperparameters: Hyperparameter[]): IClassifier
imputer_object
normalizer_object
classifier_object
1
1
1
Imputation
ION_CARRIED_FORWARD_IMPUTATION
LATION
LATION
E_IMPUTATION
IImputer
fit(df: DataFrame): void
transform(df: DataFrame): DataFrame[]
INormalizer
fit(df: DataFrame): void
transform(df: DataFrame): DataFrame
IClassifi
fit(df: DataFrame): v
predict(df: DataFram


getMostFrequentPredictionAndReset(): string
extractFeature(data: DataFrame[], feature: Feature): DataFrame
incrementPredictionCount(labelId: ObjectId): void
1
allSensorDataPoints
*
SensorDataPoints
sensor: string
1
dataPoints
*
DataPoint
timestamp: integer
values: integer[]
imputer_object
normalizer_object
classifier_object
IClassifier
ame): void
ataFrame): ndarray