Automatically Build Multiple Patient-Level Predictive Models

Jenna Reps, Martijn J. Schuemie, Patrick B. Ryan, Peter R. Rijnbeek

2021-12-16

Contents

1 Introduction

In our `paper`, we propose a standardised framework for patient-level prediction that utilizes the OMOP CDM and standardized vocabularies, and describe the open-source software that we developed implementing the framework's pipeline. The framework is the first to enforce existing best practice guidelines and will enable open dissemination of models that can be extensively validated across the network of OHDSI collaborators.

One our best practices is that we see the selection of models and all study setting as an emperical question, i.e. we should use a data-driven approach in which we try many settings. This vignette describes how you can use the Observational Health Data Sciencs and Informatics (OHDSI) `PatientLevelPrediction` package to automatically build multiple patient-level predictive models, e.g. different population settings, covariate settings, and modelsetting. This vignette assumes you have read and are comfortable with building single patient level prediction models as described in the `BuildingPredictiveModels` vignette.

Note that it is also possible to generate a Study Package directly in Atlas that allows for multiple patient-level prediction analyses this is out-of-scope for this vignette.

2 Creating a model design

The first step is to specify each model you wish to develop by using the `createModelDesign` function. This function requires the following:

Table 1: The inputs for the model design

input	Description
targetId	The id for the target cohort
outcomeId	The id for the outcome
restrictPlpDataSettings	The settings used to restrict the target population, created with createRestrictPlpDataSettings()
populationSettings	The settings used to restrict the target population and create the outcome labels, created with createStudyPopulationSettings()
covariateSettings	The settings used to define the covariates, created with FeatureExtraction::createDefaultCovariateSettings()
sampleSettings	The settings used to define any under/over sampling, created with createSampleSettings()
featureEngineeringSettings	The settings used to define any feature engineering, created with createFeatureEngineeringSettings()
preprocessSettings	The settings used to define any preprocessing, created with createPreprocessSettings()
modelSettings	The settings used to define the model fitting settings, such as setLassoLogisticRegression()

2.1 Model design example 1

For example, if we wanted to predict the outcome (id 2) occuring for the first time within 180 days of the the target population index date (id 1). We are only interested in index dates betwrrn 2018-2020. Finally, we only want to use age, gender in 5 year buckets and conditions as features. The model can be specified by:

```
# Model 1 is only using data between 2018-2020:
restrictPlpDataSettings <- createRestrictPlpDataSettings(
  studyStartDate = '20180101',
  studyEndDate = '20191231'
  )

# predict outcome within 1 to 180 days after index
# remove people with outcome prior and with < 365 days observation
populationSettings <- createStudyPopulationSettings(
  binary = T,
  firstExposureOnly = T,
  washoutPeriod = 365,
  removeSubjectsWithPriorOutcome = T,
  priorOutcomeLookback = 9999,
  requireTimeAtRisk = F,
  riskWindowStart = 1,
  riskWindowEnd = 180
)

# use age/gender in groups and condition groups as features
covariateSettings <- FeatureExtraction::createCovariateSettings(
  useDemographicsGender = T,
  useDemographicsAgeGroup = T,
  useConditionGroupEraAnyTimePrior = T
)

modelDesign1 <- createModelDesign(
  targetId = 1,
```

```
  outcomeId = 2,
  restrictPlpDataSettings = restrictPlpDataSettings,
  populationSettings = populationSettings,
  covariateSettings = covariateSettings,
  featureEngineeringSettings = createFeatureEngineeringSettings(),
  sampleSettings = createSampleSettings(),
  preprocessSettings = createPreprocessSettings(),
  modelSettings = setLassoLogisticRegression()
  )
```

2.2 Model design example 2

For the second example, we want to predict the outcome (id 2) occuring for the first time within 730 days of the the target population index date (id 1). We want to train a random forest classifier. Finally, we want to use age, gender in 5 year buckets, drug ingredients (and groups) and conditions as features. The model can be specified by:

```
# Model 2 has no restrictions when extracting data
restrictPlpDataSettings <- createRestrictPlpDataSettings(
  )

# predict outcome within 1 to 730 days after index
# remove people with outcome prior and with < 365 days observation
populationSettings <- createStudyPopulationSettings(
  binary = T,
  firstExposureOnly = T,
  washoutPeriod = 365,
  removeSubjectsWithPriorOutcome = T,
  priorOutcomeLookback = 9999,
  requireTimeAtRisk = F,
  riskWindowStart = 1,
  riskWindowEnd = 730
)

# use age/gender in groups and condition/drug groups as features
covariateSettings <- FeatureExtraction::createCovariateSettings(
  useDemographicsGender = T,
  useDemographicsAgeGroup = T,
  useConditionGroupEraAnyTimePrior = T,
  useDrugGroupEraAnyTimePrior = T
)

modelDesign2 <- createModelDesign(
  targetId = 1,
  outcomeId = 2,
  restrictPlpDataSettings = restrictPlpDataSettings,
  populationSettings = populationSettings,
  covariateSettings = covariateSettings,
  featureEngineeringSettings = createRandomForestFeatureSelection(ntrees = 500, maxDepth = 7),
  sampleSettings = createSampleSettings(),
  preprocessSettings = createPreprocessSettings(),
  modelSettings = setRandomForest()
  )
```

2.3 Model design example 3

For the third example, we want to predict the outcome (id 5) occuring during the cohort exposure of the the target population (id 1). We want to train a gradient boosting machine. Finally, we want to use age, gender in 5 year buckets and indications of measurements taken as features. The model can be specified by:

```
# Model 3 has no restrictions when extracting data
restrictPlpDataSettings <- createRestrictPlpDataSettings(
  )

# predict outcome during target cohort start/end
# remove people with  < 365 days observation
populationSettings <- createStudyPopulationSettings(
  binary = T,
  firstExposureOnly = T,
  washoutPeriod = 365,
  removeSubjectsWithPriorOutcome = F,
  requireTimeAtRisk = F,
  riskWindowStart = 0,
  startAnchor =  'cohort start',
  riskWindowEnd = 0,
  endAnchor = 'cohort end'
)

# use age/gender in groups and measurement indicators as features
covariateSettings <- FeatureExtraction::createCovariateSettings(
  useDemographicsGender = T,
  useDemographicsAgeGroup = T,
  useMeasurementAnyTimePrior = T,
  endDays = -1
)

modelDesign3 <- createModelDesign(
  targetId = 1,
  outcomeId = 5,
  restrictPlpDataSettings = restrictPlpDataSettings,
  populationSettings = populationSettings,
  covariateSettings = covariateSettings,
  featureEngineeringSettings = createFeatureEngineeringSettings(),
  sampleSettings = createSampleSettings(),
  preprocessSettings = createPreprocessSettings(),
  modelSettings = setGradientBoostingMachine()
  )
```

3 Running multiple models

As we will be downloading loads of data in the multiple plp analysis it is useful to set the Andromeda temp folder to a directory with write access and plenty of space. `options(andromedaTempFolder = "c:/andromedaTemp")`

To run the study requires setting up a connectionDetails object

```
dbms <- "your dbms"
user <- "your username"
pw <- "your password"
```

```
server <- "your server"
port <- "your port"

connectionDetails <- DatabaseConnector::createConnectionDetails(dbms = dbms,
                                                                 server = server,
                                                                 user = user,
                                                                 password = pw,
                                                                 port = port)
```

Next you need to specify the cdmDatabaseSchema where your cdm database is found and workDatabaseSchema where your target population and outcome cohorts are and you need to specify a label for the database name: a string with a shareable name of the database (this will be shown to OHDSI researchers if the results get transported).

```
cdmDatabaseSchema <- "your cdmDatabaseSchema"
workDatabaseSchema <- "your workDatabaseSchema"
cdmDatabaseName <- "your cdmDatabaseName"
cohortTable <- "your cohort table",

databaseDetails <- createDatabaseDetails(
  connectionDetails = connectionDetails,
  cdmDatabaseSchema = cdmDatabaseSchema,
  cdmDatabaseName = cdmDatabaseName ,
  cohortDatabaseSchema = workDatabaseSchema,
  cohortTable = cohortTable,
  outcomeDatabaseSchema = workDatabaseSchema,
  outcomeTable = cohortTable
  cdmVersion = 5
    )
```

Now you can run the multiple patient-level prediction analysis:

```
results <- runMultiplePlp(
  databaseDetails = databaseDetails,
  modelDesignList = list(
    modelDesign1,
    modelDesign2,
    modelDesign3
    ),
  onlyFetchData = F,
  splitSettings = createDefaultSplitSetting(),
  logSettings = createLogSettings(),
  saveDirectory =  "./PlpMultiOutput"
  )
```

This will then save all the plpData objects from the study into "./PlpMultiOutput/plpData_T1_L" and the results into "./PlpMultiOutput/Analysis_". The csv file named settings.csv found in "./PlpMultiOutput" has a row for each prediction model developed and points to the plpData and settings used for the model development, it also has descriptions of the cohorts if these are input by the user.

Note that if for some reason the run is interrupted, e.g. because of an error, a new call to `runMultiplePlp` will continue and not restart until you remove the output folder.

4 Validating multiple models

If you have access to multiple databases on the same server in different schemas you could evaluate accross these using this call:

```
validationDatabaseDetails <- createDatabaseDetails(
  connectionDetails = connectionDetails,
  cdmDatabaseSchema = 'new cdm schema',
  cdmDatabaseName = 'validation database',
  cohortDatabaseSchema = workDatabaseSchema,
  cohortTable = cohortTable,
  outcomeDatabaseSchema = workDatabaseSchema,
  outcomeTable = cohortTable,
  cdmVersion = 5
  )

val <- validateMultiplePlp(
  analysesLocation = "./PlpMultiOutput",
  valdiationDatabaseDetails = validationDatabaseDetails,
  validationRestrictPlpDataSettings = createRestrictPlpDataSettings(),
  recalibrate = NULL,
  saveDirectory = "./PlpMultiOutput/validation"
  )
```

This then saves the external validation results in the validation folder of the main study (the outputLocation you used in runPlpAnalyses).

5 Viewing the results

To view the results for the multiple prediction analysis:

```
viewMultiplePlp(analysesLocation="./PlpMultiOutput")
```

If the validation directory in "./PlpMultiOutput" has results, the external validation will also be displayed.

6 Acknowledgments

Considerable work has been dedicated to provide the `PatientLevelPrediction` package.

```
citation("PatientLevelPrediction")
```

```
##
## To cite PatientLevelPrediction in publications use:
##
## Reps JM, Schuemie MJ, Suchard MA, Ryan PB, Rijnbeek P (2018). "Design and implementation of a
## standardized framework to generate and evaluate patient-level prediction models using
## observational healthcare data." _Journal of the American Medical Informatics Association_,
## *25*(8), 969-975. <URL: https://doi.org/10.1093/jamia/ocy032>.
##
## A BibTeX entry for LaTeX users is
##
##   @Article{,
##     author = {J. M. Reps and M. J. Schuemie and M. A. Suchard and P. B. Ryan and P. Rijnbeek},
##     title = {Design and implementation of a standardized framework to generate and evaluate patient-l
##     journal = {Journal of the American Medical Informatics Association},
```

```
##     volume = {25},
##     number = {8},
##     pages = {969-975},
##     year = {2018},
##     url = {https://doi.org/10.1093/jamia/ocy032},
##   }
```

Please reference this paper if you use the PLP Package in your work:

Reps JM, Schuemie MJ, Suchard MA, Ryan PB, Rijnbeek PR. Design and implementation of a standardized framework to generate and evaluate patient-level prediction models using observational healthcare data. J Am Med Inform Assoc. 2018;25(8):969-975.