Automatically Build Multiple Patient-Level Predictive Models

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

2018-09-09

Contents

1 Introduction

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. This vignette assumes you have read and are comfortable with building single patient level prediction models as described in the `BuildingPredictiveModels` vignette.

2 Creating the setting lists

To develop multiple models the user has to create a list of Study Populations Settings, Covariate Settings, and Model Settings. These list will then be combined in a Model Analysis List and all combinations of the elements in this list will be automatically run by the package.

2.1 Study population settings

Suppose we like to make the following three population settings:

- study population 1: includes people who have the outcome but leave the database before the end of time-at-risk and only those without the outcome who are observed for the whole time-at-risk period.
- study population 2: includes people who are observed for the whole time-at-risk period.
- study population 3: includes people who do not have to be observed for the whole time-at-risk.

The create a study population setting list we use the function `createStudyPopulationSettings` as described below:

```
studyPop1 <- createStudyPopulationSettings(binary = T,
                                           includeAllOutcomes = T,
                                           removeSubjectsWithPriorOutcome = TRUE,
                                           priorOutcomeLookback = 99999,
                                           requireTimeAtRisk = T,
                                           minTimeAtRisk=364,
                                           riskWindowStart = 1,
                                           riskWindowEnd = 365,
                                           verbosity = "INFO")

studyPop2 <- createStudyPopulationSettings(binary = T,
                                           includeAllOutcomes = F,
                                           removeSubjectsWithPriorOutcome = TRUE,
                                           priorOutcomeLookback = 99999,
                                           requireTimeAtRisk = T,
                                           minTimeAtRisk=364,
                                           riskWindowStart = 1,
                                           riskWindowEnd = 365,
                                           verbosity = "INFO")

studyPop3 <- createStudyPopulationSettings(binary = T,
                                           includeAllOutcomes = F,
                                           removeSubjectsWithPriorOutcome = TRUE,
                                           priorOutcomeLookback = 99999,
                                           requireTimeAtRisk = F,
                                           minTimeAtRisk=364,
                                           riskWindowStart = 1,
                                           riskWindowEnd = 365,
                                           verbosity = "INFO")

populationSettingList <- list(studyPop1,studyPop2,studyPop3)
```

2.2 Covariate settings

The covariate settings are created using `createCovariateSettings`. We can create multiple covariate settings and then combine them in a list:

```
covSet1 <- createCovariateSettings(useDemographicsGender = T,
                                   useDemographicsAgeGroup = T,
                                   useConditionGroupEraAnyTimePrior = T,
                                   useDrugGroupEraAnyTimePrior = T)

covSet2 <- createCovariateSettings(useDemographicsGender = T,
                                   useDemographicsAgeGroup = T,
                                   useConditionGroupEraAnyTimePrior = T,
                                   useDrugGroupEraAnyTimePrior = F)

covariateSettingList <- list(covSet1, covSet2)
```

2.3 Model settings

The model settings requires running the setModel functions for the machine learning models of interest and specifying the hyper-parameter search and then combining these into a list. For example, if we wanted to try a logistic regression, gradient boosting machine and ada boost model then:

```
gbm <- setGradientBoostingMachine()
lr <- setLassoLogisticRegression()
ada <- setAdaBoost()

modelList <- list(gbm, lr, ada)
```

2.4 Model analysis list

To create the complete plp model settings use `createPlpModelSettings` to combine the population, covariate and model settings.

```
modelAnalysisList <- createPlpModelSettings(modelList = modelList,
                                    covariateSettingList = covariateSettingList,
                                    populationSettingList = populationSettingList)
```

3 Running multiple models

As we will be downloading loads of data in the multiple plp analysis it is useful to set fftempdir to a directory with write access and plenty of space. `options(fftempdir = 'T:/fftemp')`

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.

```
cdmDatabaseSchema <- "your cdmDatabaseSchema"
workDatabaseSchema <- "your workDatabaseSchema"
```

Now you can run the multiple patient-level prediction analysis by specifying the target cohort ids and outcome ids

```
allresults <- runPlpAnalyses(connectionDetails = connectionDetails,
                             cdmDatabaseSchema = cdmDatabaseSchema,
                             oracleTempSchema = cdmDatabaseSchema,
                             cohortDatabaseSchema = workDatabaseSchema,
                             cohortTable = "your cohort table",
```

```
                            outcomeDatabaseSchema = workDatabaseSchema,
                            outcomeTable = "your cohort table",
                            cdmVersion = 5,
                            outputFolder = "./PlpMultiOutput",
                            modelAnalysisList = modelAnalysisList,
                            cohortIds = c(2484,6970),
                            cohortNames = c('visit 2010','test cohort'),
                            outcomeIds = c(7331,5287),
                            outcomeNames =  c('outcome 1','outcome 2'),
                            maxSampleSize = NULL,
                            minCovariateFraction = 0,
                            normalizeData = T,
                            testSplit = "person",
                            testFraction = 0.25,
                            splitSeed = NULL,
                            nfold = 3,
                            verbosity = "INFO")
```

This will then save all the plpData objects from the study into "./PlpMultiOutput/plpData", the populations for the analysis into "./PlpMultiOutput/population" and the results into "./PlpMultiOutput/Result". The csv named settings.csv found in "./PlpMultiOutput" has a row for each prediction model developed and points to the plpData and population used for the model development, it also has descriptions of the cohorts and settings 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 `RunPlpAnalyses` 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:

```
val <- evaluateMultiplePlp(analysesLocation = "./PlpMultiOutput",
                           outputLocation = "./PlpMultiOutput/validation",
                           connectionDetails = connectionDetails,
                           validationSchemaTarget = list('new_database_1.dbo',
                                                         'new_database_2.dbo'),
                           validationSchemaOutcome = list('new_database_1.dbo',
                                                          'new_database_2.dbo'),
                           validationSchemaCdm = list('new_database_1.dbo',
                                                      'new_database_2.dbo'),
                           databaseNames = c('database1','database2'),
                           validationTableTarget = 'your new cohort table',
                           validationTableOutcome = 'your new cohort table')
```

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")
```

```
##
##   Jenna Reps, Martijn J. Schuemie, Marc A. Suchard, Patrick B.
##   Ryan and Peter R. Rijnbeek (2018). PatientLevelPrediction:
##   Package for patient level prediction using data in the OMOP
##   Common Data Model. R package version 2.0.5.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {PatientLevelPrediction: Package for patient level prediction using data in the OMOP
## Common Data Model},
##     author = {Jenna Reps and Martijn J. Schuemie and Marc A. Suchard and Patrick B. Ryan and Peter R
##     year = {2018},
##     note = {R package version 2.0.5},
##   }
```

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.