NHS
Nottinghamshire Healthcare
NHS Foundation Trust

Scaling Up Shiny and Text Mining for National Health Decisions

Andreas D Soteriades, Data Scientist (andreas.soteriades@nottshc.nhs.uk)

Chris Beeley, Senior Data Scientist (chris.beeley@nottshc.nhs.uk)

Clinical Development Unit, Data Science Team, Nottinghamshire Healthcare NHS Foundation Trust

Background

NHS trusts get loads of feedback from patients

Content

- What do patients talk about?
- Are they happy/unhappy?
- How can we surface this information and convert it into actionable insights for managers?

Labelling

- Specialized staff read and label the text (e.g. Access, Communication, Environment/ facilities etc.)
- A resource-consuming process

Background

NHS trusts get loads of feedback from patients

Content

- What do patients talk about?
- Are they happy/unhappy?
- How can we surface this information and convert it into actionable insights for managers?

Labelling

- Specialized staff read and label the text (e.g. Access, Communication, Environment/ facilities etc.)
- A resource-consuming process

Text Mining – sentiment analysis, TF-IDFs, network diagrams

Text Classification – semi-automate the labelling of unlabelled text

Background

Nottinghamshire Healthcare
NHS Foundation Trust

NHS trusts get loads of feedback from patients

Content

- What do patients talk about?
- Are they happy/unhappy?
- How can we surface this information and convert it into actionable insights for managers?

Labelling

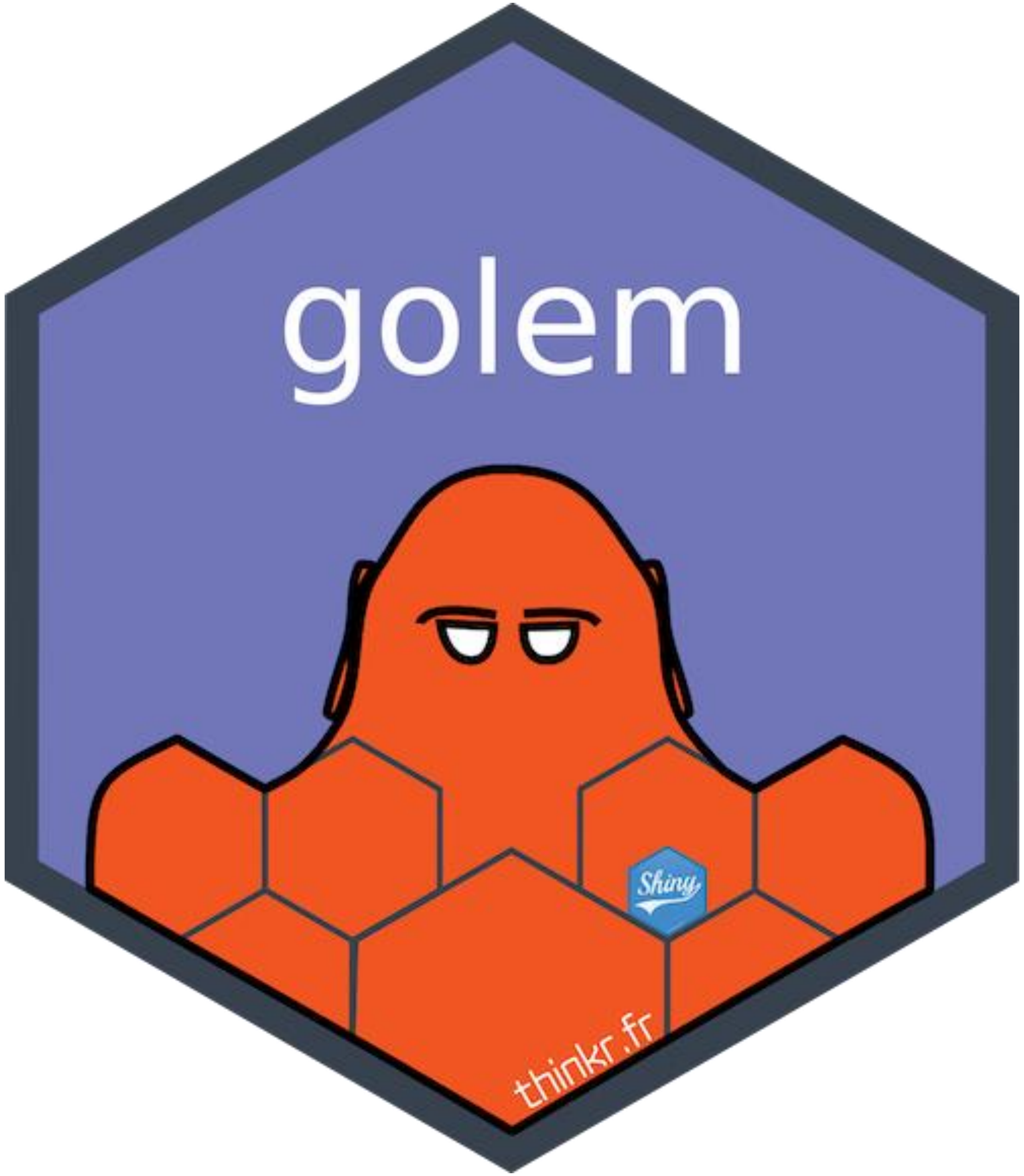
- Specialized staff read and label the text (e.g. Access, Communication, Environment/ facilities etc.)
- A resource-consuming process

Text Mining – sentiment analysis, TF-IDFs, network diagrams

Text Classification – semi-automate the labelling of unlabelled text

Not just for our trust

- Free, open-source **licensed** solutions
- Built with user groups
- Test and roll-out trusts



Unclassified

Shiny before {golem}

Nottinghamshire Healthcare
NHS Foundation Trust

- Enormous (2000+ line) server.R files
- Difficult to debug
- Difficult to test
- Difficult to collaborate on
- Difficult to deploy
- Difficult to use with different data

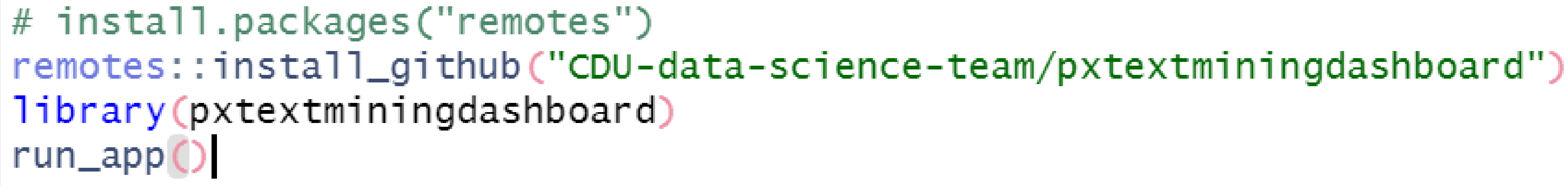
A Shiny app should be…

- Modular
- Strict as to where the business logic is- or isn't
- Documented (the functions, not the app)
- Tested
- Shareable
- Ideally, agnostic to deployment

Shiny with {golem}

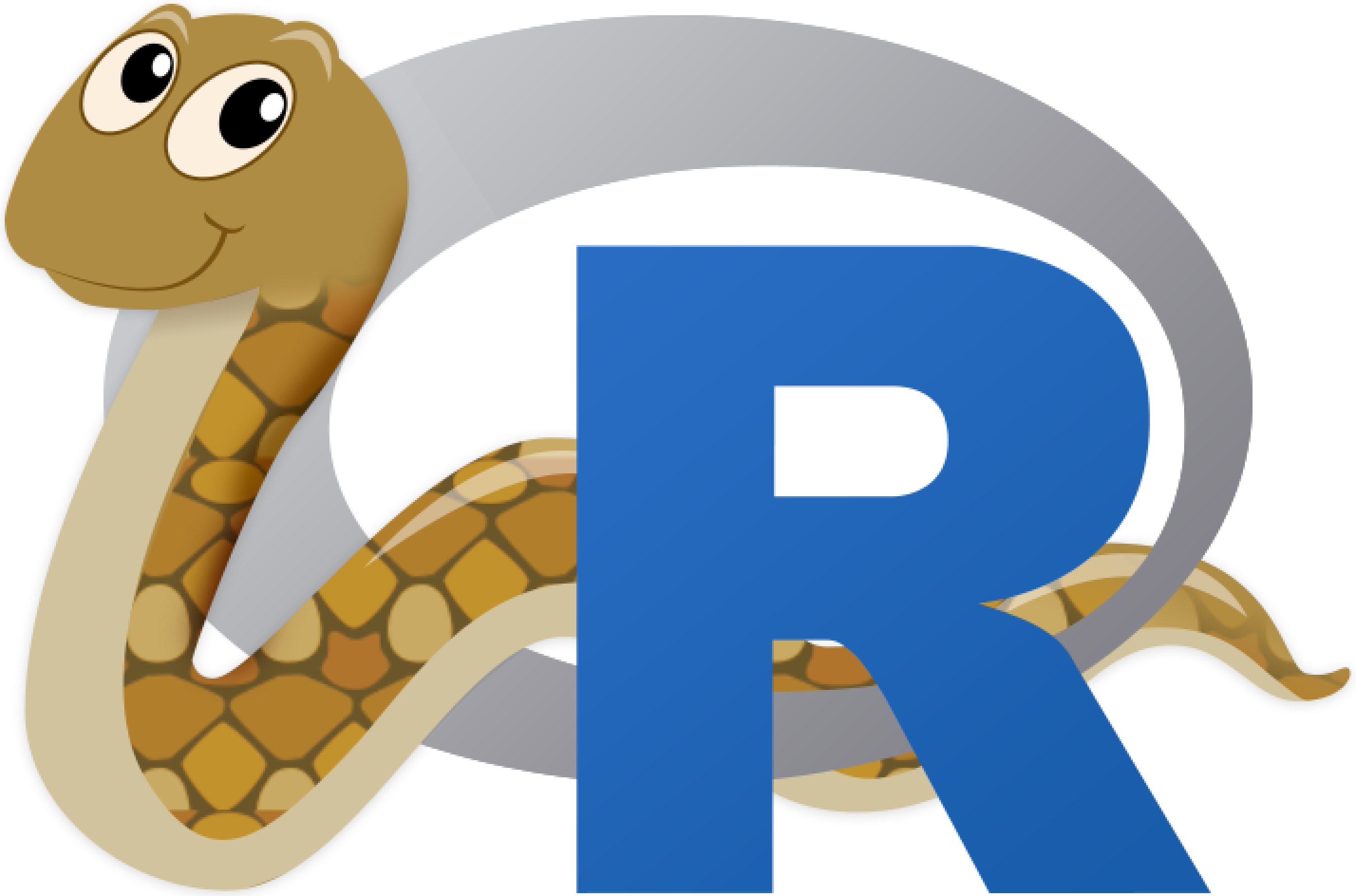
Nottinghamshire Healthcare
NHS Foundation Trust

- *{golem} is "[…] an opinionated framework for building production-grade shiny applications"*
- In {golem}, all Shiny applications are R packages
- Packages make it easy to test, manage dependencies, and deploy



```
# install.packages("remotes")
remotes::install_github("CDU-data-science-team/pxtextminingdashboard")
library(pxtextminingdashboard)
run_app()
```

- Beautiful!


R

Text Classification

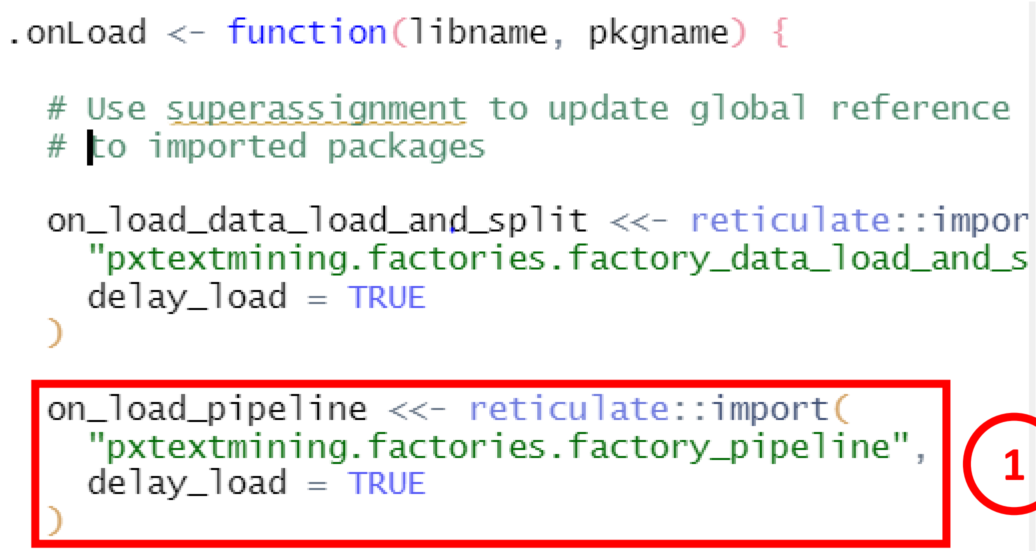
Nottinghamshire Healthcare
NHS Foundation Trust

- Much of the NHS is R-oriented
- When it comes to Machine Learning
  - *Python is particularly well suited for deploying machine learning at a large scale* (IBM, accessed Aug 2021)
  - Python is much faster than R (e.g. Towards Data Science, accessed Aug 2021)
- Our approach
  - Harness the advantages of Python
  - Make a Python library (pxtextmining)
  - Make a R wrapper (pxtextmineR)

Scikit-learn & its R counterparts

	R		Python
	tidymodels	**mlr3**	**Scikit-learn**
ML models	All three libraries offer an interface for integrating and standardizing the use of different models. The problem with R though is that it borrows models from different packages, the quality of which depends on the authors' skills and individual efforts (e.g. willingness to make models faster, add more features or actively maintain package). In Scikit-learn, models in it are built *for* it.		
Speed	I am not aware of a benchmarking exercise. But mlr3 could be faster and more efficient, because it uses data.table and R6 objects. Speed also strongly depends on the individual packages that tidymodels and mlr3 borrow the ML models from.		Designed to be fast. Interoperates with NumPy and SciPy for fast scientific computing. Many core algorithms built in Cython.
User-friendliness	tidyverse-style use of "%>%". Loads of resources. More appropriate for newbies.	Resources constantly updated. More "hardcore" ML in its looks, but easy to catch up if familiar with ML, Python and "classes".	Solid, well-organized and consistent User Guide that also covers model theory and has numerous examples.
Text classification	Possible, with a few models available for Bag of Words (BoW) learning- although I do not know how fast.	Currently at early stage and slow.	Large collection of mind-blowingly fast models for BoW.

{reticulate} in action: pxtextmineR



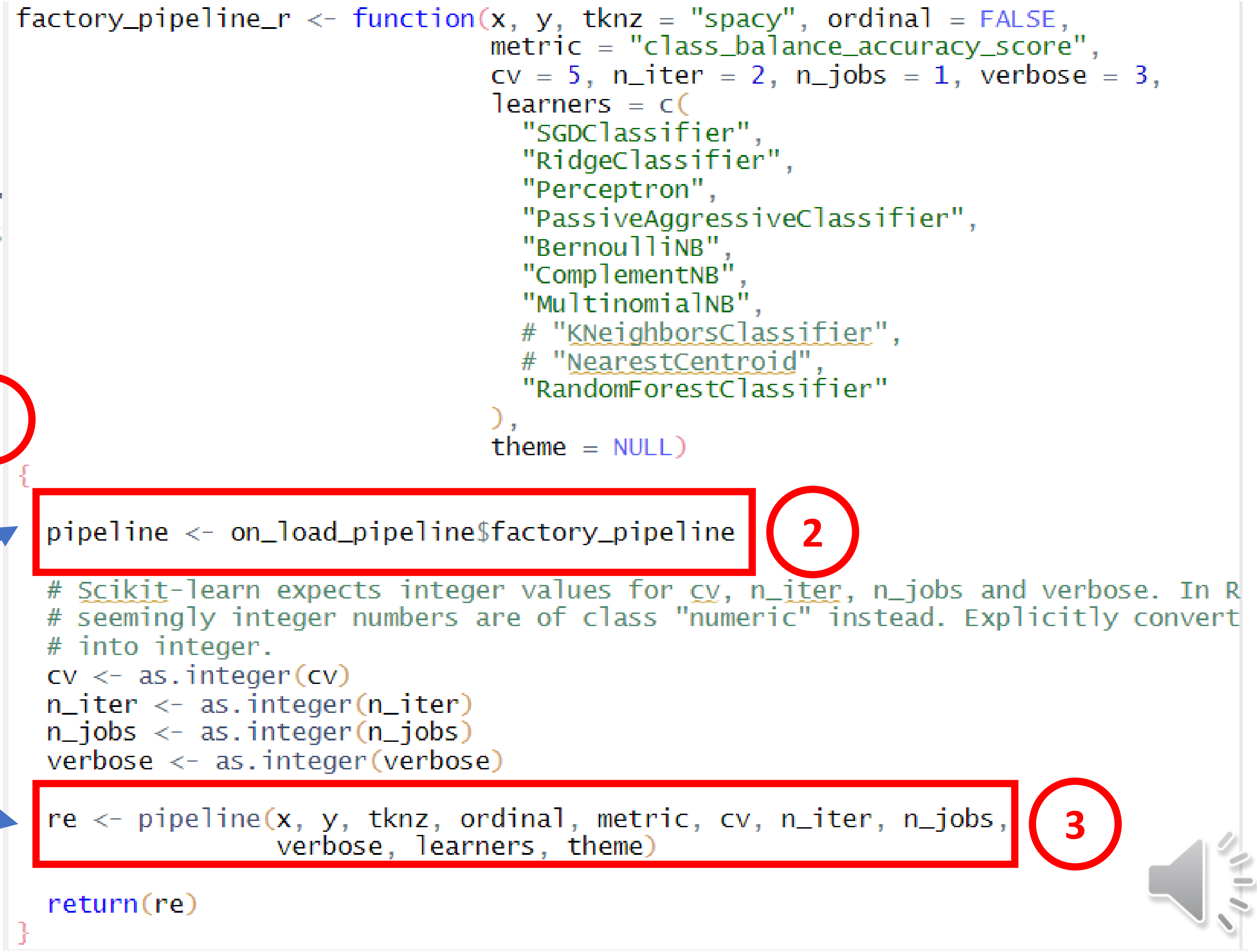
```
.onLoad <- function(libname, pkgname) {

  # Use superassignment to update global reference
  # to imported packages

  on_load_data_load_and_split <<- reticulate::impor
    "pxtextmining.factories.factory_data_load_and_s
    delay_load = TRUE
  )

  on_load_pipeline <<- reticulate::import(
    "pxtextmining.factories.factory_pipeline",
    delay_load = TRUE
  )
```

1



```
factory_pipeline_r <- function(x, y, tknz = "spacy", ordinal = FALSE,
                               metric = "class_balance_accuracy_score",
                               cv = 5, n_iter = 2, n_jobs = 1, verbose = 3,
                               learners = c(
                                 "SGDClassifier",
                                 "RidgeClassifier",
                                 "Perceptron",
                                 "PassiveAggressiveClassifier",
                                 "BernoulliNB",
                                 "ComplementNB",
                                 "MultinomialNB",
                                 # "KNeighborsClassifier",
                                 # "NearestCentroid",
                                 "RandomForestClassifier"
                               ),
                               theme = NULL)
{
  pipeline <- on_load_pipeline$factory_pipeline

  # Scikit-learn expects integer values for cv, n_iter, n_jobs and verbose. In R
  # seemingly integer numbers are of class "numeric" instead. Explicitly convert
  # into integer.
  cv <- as.integer(cv)
  n_iter <- as.integer(n_iter)
  n_jobs <- as.integer(n_jobs)
  verbose <- as.integer(verbose)

  re <- pipeline(x, y, tknz, ordinal, metric, cv, n_iter, n_jobs,
                 verbose, learners, theme)

  return(re)
}
```

2

3

This is where all the magic happens!

In a nutshell

Our toolkit

- We **love** R RStudio golem R !
- We have the tools to scale up

Our vision

- Data Science for the betterment of a public service
- From our trust to the whole of the NHS

So far

- We’ve proved the concept

What’s next

- Make nationwide impact
- Deep Learning!

Thank you!

pxtextmining

https://pypi.org/project/pxtextmining/

https://github.com/CDU-data-science-team/pxtextmining

pxtextmineR

https://github.com/nhs-r-community/pxtextmineR

Dashboards with Golem

https://github.com/CDU-data-science-team/pxtextminingdashboard

https://github.com/CDU-data-science-team/experiencesdashboard

experienceAnalysis

https://github.com/CDU-data-science-team/experienceAnalysis

Shiny/Golem

https://engineering-shiny.org/

https://thinkr.fr/

https://mastering-shiny.org/

reticulate

https://rstudio.github.io/reticulate/

Us!

https://cdu-data-science-team.github.io/team-blog/

Funding

NHS England

NHS-R COMMUNITY