Capstone Project 2 : Clinical Trials - Harvard Professional Certificate in Data Science

Vincent AKIKI

February 2022

Contents

1 INTRODUCTION

The focus of this report is to create a predictive classification model capable of detecting cancer related clinical trials from the International Clinical Trials Registry Platform (**ICTRP**) founded by the World Health Organization (**WHO**).

This is performed by extracting as much insight as possible from the **clinical trials text** using text mining methods going from **tokenization** to **parsing** and **matrices factorization**. These methods were applied on **cancer** tagged clinical trials texts and represents the basis of our **train** and **test** machine learning datasets.

The **predictive model** built is an ensembles of decision trees, a **Random Forest** algorithm. The performance of this automated classifier has a **96.85%** degree of **accuracy** with a **specificity** of **98.65%** (the percentage of "Non-Cancer" trials correctly predicted among "Non-Cancer") and a **sensitivity** of **91.33%** (the percentage of "Cancer" trials correctly predicted among "Cancer") .

All codes used for this report are provided at the end in the appendix

2 METHODS & ANALYSIS

2.1 Data collection & structure

The International Clinical Trials Registry Platform (**ICTRP**) get its data from **17 partner registries** and data providers, like the "EU Clinical Trials Register (EU-CTR)", "ClinicalTrials.gov " or "Japan Primary Registries Network (JPRN)".

Our ICTRP dataset comes from an **Xml** generated in august 2021 from the **ICTRP advanced search web interface** (*https://trialsearch.who.int/AdvSearch.aspx*) by selecting trials registered between the 1st January 2018 and 30th June 2018. No further filtering was applied, this Xml includes not only interventional, but also registered observational, patient registry or expanded access studies.

2.2 Data wrangling

To be able to analyse and run machine learning algorithms, our data need to be **extracted**, **cleansed**, and **reshaped**.

We start by collecting the needed items and storing them into a table format. Then checking for missing data, deleting duplicates.

Followed by selecting interventional studies and tagging them with the "Cancer" and "Non-Cancer" labels

Step 1 : call the source file (Xml file)

The **Xml** generated in august 2021 from the **ICTRP advanced search web interface** is stored and called from my github repository.

```
# Get from my github repository the Xml file downloaded from ICTRP (12 586 trials)
temp <- tempfile()
download.file(
  "https://github.com/vakiki/Clinical-Trials-Project/raw/main/ICTRP-Results_mi2018.zip"
  ,temp)
dlfile <- read_xml(unz(temp, "ICTRP-Results_mi2018.xml"))
unlink(temp)
```

Step 2 : select the wanted items and stores them in table format

There are currently **24 items** in the WHO trial registration database, e.g. "date of registration" ; "source of funding". Since our challenge here is to be able to identify cancer related trials among all the trials, not all these items are useful. In fact, only the following information from the ICTRP Xml were selected :

1. **ICTRP_TrialID** [Internal_Number] : unique ICTRP identifier generated when a trial is uploaded from a registry for the first time.
2. **Original_TrialID** [TrialID] : unique identifier from the issuing authority (e.g. NCT, ISRCTN, ACTRN).
3. **Study_Source** [Source_Register] : name of the partner registries and data providers among the list of 17.
4. **Study_type** : nature of the clinical study. It includes interventional studies (also called clinical trials), observational studies (including patient registries), and expanded access.
5. **Public_title** : title intended for the lay public in easily understood language.
6. **Scientific_title** : scientific title of the study as it appears in the protocol submitted for funding and ethical review. Include trial acronym if available.
7. **Study_Condition** : primary health conditions and the disease, disorder, syndrome, illness, or injury that is being studied (e.g., stroke, breast cancer, medication error).

This ICTRP dataset contains **12586 records** with one row per primary ID.

The **ICTRP_TrialID** is used to identify the trials, the **Original_TrialID** and the **Study_Source** among the **Public_title** and the **Scientific_title** are used to eliminates duplicates, the **Study_type** is used to select only the interventional studies and the **Public_title** ; the **Scientific_title** and the **Study_Condition** are used to provide the text on which our Natural Language Processing (**NLP**) and **supervised learning classification** will be performed to detect whether the study fall into the "Cancer" or the "Non-Cancer" category.

Step 3 : check for missing data and delete duplicates

Missing data as well as **duplicated data** present various problems. They can cause bias in the estimation of parameters and can reduce the representativeness of the samples.

It is important to check if our dataset do not contain a significant amount of missing data and to be able to guarantee a clean dataset by detecting and deleting duplicates.

```
# Check for missing values in our dataset
length(which(!complete.cases(ICTRP_table)))
```

```
## [1] 0
```

We have **0 missing data** in our dataset.

Since ICTRP aggregates data from various partner registries, the ICTRP dataset contains many internal duplicates. As said above, we can rely on **Original_TrialID** ; **Public_title** and **Scientific_title** items to identify and delete probable duplicates.

```
# Delete duplicates within clinical trials records

# 1/ Based on the "Original trial ID"
ICTRP_table <- distinct(ICTRP_table, Original_TrialID, .keep_all = TRUE)
# 2/ Based on the "Scientific title" text
ICTRP_table <- distinct(ICTRP_table, Scientific_title, .keep_all = TRUE)
# 3/ Based on the "Public title" text
ICTRP_table <- distinct(ICTRP_table, Public_title, .keep_all = TRUE)
```

Our dataset has almost 1 duplicated study over 6 studies (**16.9712%**). Our de-duplicated dataset has now **10450 records**.

Step 4 : select only interventional studies

Table 1, show the study type distribution of our dataset.

Table 1: Study Type initial categories distribution

Study_type	n
BA/BE	8
Basic science	19
Cause	20
Cause/Relative factors study	5
Diagnostic test	75
Epidemilogical research	17
Health services reaserch	3
Intervention	162
interventional	954
Interventional	5040
INTERVENTIONAL	40
Interventional clinical trial of medicinal product	763
Interventional study	764
Interventional Study	114
observational	112
Observational	1471
OBSERVATIONAL	4
Observational [Patient Registry]	206
Observational study	318
Observational Study	36
Others,meta-analysis etc	3
PMS	13
Prevention	4
Prognosis study	17
Relative factors research	225
Screening	10
Treatment study	46
Unknown	1

Most of the time we are interested by **interventional** trials rather than other types of studies as observational studies. In fact, an interventional trial is specifically designed to evaluate the direct impacts of treatment or preventative measures on disease.

To mimic at best our future use of our present machine learning algorithm, we will train and test it on interventional studies.

To keep only interventional studies we need to select the following study_type categories : "Intervention", "Interventional", "Interventional Clinical Trial Of Medicinal Product", "Interventional Study".

Our dataset is reduced now to only interventional studies which represents **7837trials**.

Step 5 : add the "Cancer / Non-Cancer" labels

The last step before starting the NLP and Machine Learning process is to label each of our 7837 trials as "Cancer" or "Non-Cancer". This task was supported by a list of general and specified cancer keywords. These keywords were validated by two oncologist and a Quality Control was conducted on the final trials labelling.

It is a crucial step because our supervised learning performance depends also on the reliability of this labelling.

2.3 Data exploration

Here are the Cancer/Non-Cancer label distribution of our dataset :

Table 2: Cancer/Non-Cancer label distribution

Var1	Freq
Cancer	0.246
Non-Cancer	0.754

Almost 1 out of 4 trials is tagged as "Cancer".

Let's examine now if there is a difference in text length between "Cancer" and "Non-Cancer" trials.

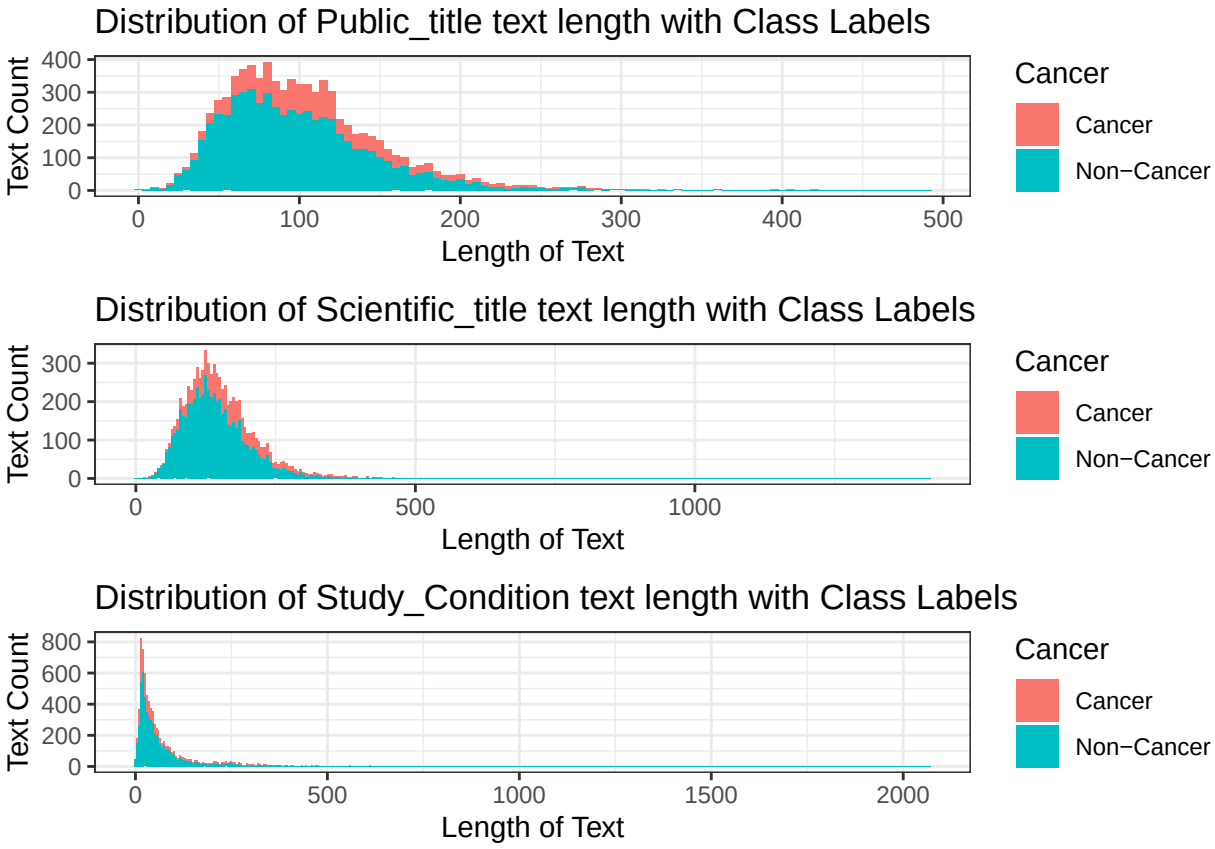

Figure 1: Text length distributions

We learn from **Figure 1**, that there is no particular difference in text length distribution between Cancer and Non-cancer trials and this among the public title text, the scientific title text and the study condition text.

2.4 Data preparation for machine learning simulations

We keep from our previous dataset the following features :

- **ID** : The original ICTRP_TrialID item.
- **Text** : Text concatenation of Public_title ; Scientific_title and Study_Condition items.
- **Pub_textLength** : The trial public title text length.
- **Sci_textLength** : The trial scientific title text length.
- **Cond_textLength** : The study condition title text length.
- **Label** : Cancer/Non-Cancer labels.

PS: the text length features are taken in consideration only for Model 3 (see further).

Step 1 : create a 70/30 percents stratified split dataset

Tokenization is the first step of natural language text preparation. The major goal of this task is to convert a stream of characters into a stream of processing units called **tokens**. Everything between whitespaces is a token. The caret package (Classification and Regression Training) provides some functionalities for splitting our data into **train** and **test** dataset to verify how good is our model.

The next step is to create a **70%/30% stratified split** from our previous dataset :

1. **train set**, that contains **70%** of the **ICTRP_table** data (**5487trials**) on which we will build and tune our machine learning model.
2. **test set**, that contains the remaining **30%** of the data (**2350trials**), which will serve to evaluate the performance of our model.

This is known as stratified split because it keeps the same dataset distribution when splitting. It preserves the 24.5% Cancer / 75.5% Non-Cancer distribution of our dataset previously noticed :

Table 3: Train Cancer/Non Cancer distribution

Var1	Freq
Cancer	0.2457
Non-Cancer	0.7543

Table 4: Test Cancer/Non Cancer distribution

Var1	Freq
Cancer	0.2455
Non-Cancer	0.7545

Step 2 : tokenize and pre-process the train text

Tokenization is the first step of natural language text preparation. The major goal of this task is to convert a stream of characters into a stream of processing units called **tokens**. Everything between whitespaces is a token.

Example :

- Text before tokeniation :
  "Phase 2 study of Crizotinib in lobular breast cancer or diffuse gastric cancer."
- Text after tokenzation :
  [Phase] [2] [study] [of] [Crizotinib] [in] [lobular] [breast] [cancer] [or] [diffuse] [gastric] [cancer] [.]

=> 14 tokens.

Because we are dealing with the dirtiest of the dirty data "free-form text", **pre-processing** is a major part of text analytics.

The examination of the text tokenization in the example above , shows that there is a lot of unhelpful tokens or what it is commonly called noises. In our study, these noises can be reduced or removed by :

1. converting all characters to **lowercase** (e.g. If vs if).
2. removing **punctuations** marks (e.g., ", ?, !).
3. removing **stop words** (e.g., the, an, a). Stop words allow the language to flow, but strictly speaking they do not add any semantics or predictive power.
4. removing **numbers** (e.g., 0, 56, 109). We are referring to digits.
5. removing **symbols** (e.g., <, @).
6. reducing words to their **stem** or root (e.g., ran, run, runs, running). It normalizes tokens that look similar and convey the same meaning into a single canonical form (e.g. run).

Example :

- Text after tokenzation :
  [Phase] [2] [study] [of] [Crizotinib] [in] [lobular] [breast] [cancer] [or] [diffuse] [gastric] [cancer] [.]
- Text after tokenzation and pre-processing :
  [phase] [studi] [crizotinib] [lobular] [breast] [cancer] [diffus] [gastric] [cancer]

=> 9 tokens.

Step 2-bis : add bi-grams (applied for Model 4 and 5)

Our previous tokens representations was built on single terms (**unigrams**). But we can consider more than a single term like **bigrams** (2 terms), trigrams (3 terms), etc. They allow us to include word ordering.

They are referred in general to "**N-grams**".

If we consider both unigrams and bi-gramsHere our previous example will give this :

- Text after tokenzation and pre-processing :
  [phase] [studi] [crizotinib] [lobular] [breast] [cancer] [diffus] [gastric] [cancer] [phase_studi] [studi_crizotinib] [crizotinib_lobular] [lobular_breast] [breast_cancer] [cancer_diffus] [diffus_gastric] [gastric_cancer] [cancer_phase_studi]

=> 18 tokens (2x9).

Note that adding N-grams to a Uni-grams presentation, like bi-grams for example, will more than double the total size of our tokens and we probably might experiment the **curse of dimensionality** problem.

Step 3 : create the "bag of words" model => "Document Feature Matrix" (DFM)

With tokenization complete, it is possible to construct a data frame where :
- each **row** represents a trial;
- each **column** represents a distinct token;
- each **cell** represents the token frequency for a document (trial).

This representation is referred to **Document-Frequency Matrix** (**DFM**).

The DFM of our previous **unigram** example will be :

docs (trials)	phase	studi	crizotinib	lobular	breast	cancer	diffus	gastric
Trial 1 text	1	1	1	1	1	2	1	1

Step 4 : enhance DFM representation with TF-IDF calculation

DFM is a very useful representation but there are some issues :
- longer documents (trials text) will tend to have higher term counts.
- terms that appear frequently across the corpus (trials) are not as important.

We can improve upon this representation if we can achieve the following :
- **normalize** documents based on their length (**TF: Term Frequency**).
- **penalize** terms that occur frequently across the corpus (**IDF : Inverse Document Frequency**).

Term Frequency : $TF(t,d) = \frac{freq(t,d)}{\sum_i^n freq(t_i,d)}$

Where :
- freq(t,d), is the count of the instances of the term t in document d;
- TF(t,d), is the proportion of the count of term t in document d;
- n, is the number of distinct terms in document d.

Inverse Document Frequency : $IDF(t,d) = log(\frac{N}{count(t)})$

Where :
- N, is the count of distinct documents in the corpus;
- count(t), is the count of documents in the corpus in which the term t is present.

So if a term shows up in every document (trial) of the corpus (all the trials) we will have : $\frac{N}{count(t)} = 1 => log(1) => 0$. In this extreme case, this term is penalized at the maximum ($IDF(t) = 0$).

To enhance our bag-of-words model we combine TF and IDF and calculate for each cell of our document-term frequency matrix the following: $TF - IDF(t,d) = TF(t,d) * IDF(t)$

Step 5 : perform a Singular Value Decomposition (SVD) on the TF-IDF matrix

Even with just using unigrams pre-process tokens as features for our machine learning model, we get a high number of features. Our **TF-IDF matrix** based on **unigrams** (Model 1) has a dimension of **5487 trials** (rows) and **14338 features** (columns).

But if we examine the cells values of our TF-IDF matrix, we find out that **most of them are equal to 0**. We are dealing with a huge flat matrix with a **sparsing of almost 99.9%** (lots of 0). We can perform a **Latent Semantic Analysis** (**LSA**) using the **Singular Value Decomposition** (**SVD**) which is a matrix factorization technic that allows us to implement **features reduction** and remediate the curse of dimensionality problem. We can therefore use more robust algorithm in our machine learning like the **Random Forest**.

Not only by selecting a fraction of the most important singular values, **SVD** dramatically reduce dimensionality, but by combining columns this matrix factorization enriches our **data signal** comparing to a highly sparsed matrix.

As with **TF-IDF** the use of **SVD** will require that new data be transformed (projected into the semantic space) before predictions can be made (machine learning). But keep in mind that this reduced factorized matrix is an approximate of our original matrix :

$\hat{d} = \sum^{-1} U^T d$

Where :
- $\hat{d}$, represents our transformed trial text on which we will create our prediction on;
- d, represents the TF-IDF projected trial text (the one that been tokenized, stemmed, stop word removed, lower cased, and TF-IDF calculated).

With **SVD** we get the terms perspective and the document (trials text) perspective at the same time.

If we transpose our **TF-IDF** matrix we will flip from a "document – term" matrix to a "term – document" matrix. This is the final representation that we want to feed our machine learning.

irlba package allows us to perform a truncated SVD. By tuning the "nv" option, we can fix the number of right singular vectors to estimate. That will give us the vectors representation of the higher-level concept extracted per trial text basis.

In our model 1, 3, 4 and 5 (see further) we selected the **300** most important and significant approximated right singular vectors out of all of the extracted terms that we can create using SVD. And for model 4 we selected the **500 most** important ones. In model 1 per example, with SVD transformation we go from **14338 features** (tokens) with lots of 0 cells to **300 features** (right singular vectors "v").

Notice that we are now dealing with what is referred to **Blackbox technic** : we gave up the possibility to explain what is going on, to gain more power. Indeed, v1,v2,..v300 values are no longer meaningful to us as the tokens were. But the v columns representation is a much better representation, it allows to perform random forest algorithm effectively because we have a much more compact representation.

Step 6 : perform our previous pre-processing pipeline on the test data

Everything that we have done so far with the **train** data, must be done on the **test** data.

We first **tokenize**, **pre-process**, and transform into a **DFM** matrix our test data. But before **calculating TF-IDF**, we must make sure that our test DFM matrix look exactly like the train DFM. This means we have to ensure that our test DFM matrix has the same feature as the train TF-IDF matrix (same tokens in the same order), otherwise our prediction will not be valid.

In text analytics, training is performed on historical set of data. That historical set of data has a collection of words that reflects the data on a specific time. It is expected with our 70%/30% split data, that each time we get a different number of features because each time we get a different combination of trials list to train and trials list to test du to our sampling process.

We must transform the actual test data and make it look the same as the actual train data. To do so, we **enforce** on this **test DFM** the **same structure** than our **train DFM** we got in the previous steps. That why we previously kept the original IDF train values, so we implement them as the test IDF features.

Step 7 : project the test TF-IDF matrix using the SVD matrix factorization projection

After applying TF-IDF on our test data that has now the same structure than our train DFM, we projected it into the same semantic space of our train data (nv = 300 models 1,3,4,5 and nv= 500 model 2) using the approximation formula : $\hat{d} = \sum^{-1} U^T d$

Step 8 : perform machine learning

The train and test features are ready to apply in our machine learning. We will leverage **Cross Validation** (**CV**) as the basis of our modelling process. **CV** is a resampling method that uses different portions of the data to test and train a model on different iterations. It is a simple, robust technique and generally results in a less biased estimate of the model that being train and test.

We will perform here a **10-fold CV repeated 3 times** (30 sampling process). We are also tuning our **Random Forest** algorithm to try **7 different values** of the number of variables randomly sampled as candidates at each split (mtry parameter). There are many other parameters, but **mtry** is perhaps the most likely to have the biggest effect on the final accuracy.

By default, a Random Forest algorithm leverages 500 trees (caret package). It means that we will be building : (10 x 3 x 7 x 500) + 500 = 105 500 trees. Caret will build one final model at the end of the process with the best **mtry** value over all the training data.

Due to this huge number of trees to build, if you run the corresponding code it will last too long. Actually it takes more than 7 hours using 3 cores on a 4 cores processor. It is not ran again here, feel free to download and run the 5 machines learning models tested in this report, from my github repository : *https://github.com/vakiki/Clinical-Trials-Project*.

3 RESULTS

3.1 Problem definition

We modeled here our automated cancer trials identifier using two types of features : the "**v vectors**" and the **text length**.

SVD "**v vectors**" features were generated from **unigrams** and/or **bigrams** text tokens from the concatenated Public_title ; Scientific_title and Study_Condition texts. The **300** or **500** most important and significant v vectors were kept for our machine learning.

The **text length** predictors or features are the **triplet variables** : "Public_title" text length ; "Scientific_title" text length and "Study_Condition" text length. The following 5 models were tested :

Predictive Model #	"v" from unigrams ?	"v" from bigrams ?	300 v features ?	500 v features ?	text length ?	Number of features	M.L. algorithm
Model 1	X		X			300	Random Forest
Model 2	X			X		500	Random Forest
Model 3	X		X		X	303	Random Forest
Model 4		X	X			300	Random Forest
Model 5	X	X	X			300	Random Forest

3.2 Model 1: 300 v features from unigrams

The first model tested had 300 v vectors generated from unigrams tokens.

Model 1 confusion matrix:

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Cancer Non-Cancer
##   Cancer        527         24
##   Non-Cancer     50       1749
##
##                Accuracy : 0.9685
##                  95% CI : (0.9606, 0.9752)
##     No Information Rate : 0.7545
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9137
##
##  Mcnemar's Test P-Value : 0.003659
##
##             Sensitivity : 0.9133
##             Specificity : 0.9865
##          Pos Pred Value : 0.9564
##          Neg Pred Value : 0.9722
##              Prevalence : 0.2455
##          Detection Rate : 0.2243
##    Detection Prevalence : 0.2345
##       Balanced Accuracy : 0.9499
##
##        'Positive' Class : Cancer
##
```

Model 1 statistics:

```
## Random Forest
##
## 5487 samples
##  300 predictor
##    2 classes: 'Cancer', 'Non-Cancer'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 4939, 4938, 4938, 4939, 4938, 4938, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##     2   0.9004920  0.6890108
##    51   0.9686500  0.9138452
##   101   0.9677394  0.9116587
##   151   0.9667670  0.9090404
##   200   0.9661585  0.9074172
##   250   0.9654906  0.9055415
##   300   0.9635471  0.9004410
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 51.
```

3.3 Model 2: 500 v features from unigrams

This predictive model is similar model 1 but with **500 v** instead of 300 v.

Model 2 confusion matrix:

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Cancer Non-Cancer
##   Cancer        528         26
##   Non-Cancer     49       1747
##
##                Accuracy : 0.9681
##                  95% CI : (0.9602, 0.9748)
##     No Information Rate : 0.7545
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9127
##
##  Mcnemar's Test P-Value : 0.01107
##
##             Sensitivity : 0.9151
##             Specificity : 0.9853
##          Pos Pred Value : 0.9531
##          Neg Pred Value : 0.9727
##              Prevalence : 0.2455
##          Detection Rate : 0.2247
##    Detection Prevalence : 0.2357
##       Balanced Accuracy : 0.9502
##
##        'Positive' Class : Cancer
##
```

Model 2 statistics:

```
## Random Forest
##
## 5487 samples
##  500 predictor
##    2 classes: 'Cancer', 'Non-Cancer'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 4939, 4938, 4938, 4939, 4938, 4938, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##     2   0.8224893  0.3657191
##     5   0.9208416  0.7607214
##    12   0.9601471  0.8872683
##    31   0.9671927  0.9091008
##    79   0.9680427  0.9122351
##   199   0.9662204  0.9075643
##   500   0.9616654  0.8953368
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 79.
```

As shown above, we got almost the same predictive power as model 1. Probably due to the fact that the **mtry** is too low (52 for model 1). Moreover, Model 2 have plus 200 more features to compute than model 1. It consumes much more resource than the models with 300 v. Almost 12 hours of computation instead of 7 hours comparing to model 1.

3.4 Model 3: 300 v features from unigrams and 3 text length features

Model 3 confusion matrix:

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Cancer Non-Cancer
##   Cancer        528         24
##   Non-Cancer     49       1749
##
##                Accuracy : 0.9689
##                  95% CI : (0.9611, 0.9756)
##     No Information Rate : 0.7545
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9149
##
##  Mcnemar's Test P-Value : 0.00497
##
##             Sensitivity : 0.9151
##             Specificity : 0.9865
##          Pos Pred Value : 0.9565
##          Neg Pred Value : 0.9727
##              Prevalence : 0.2455
##          Detection Rate : 0.2247
##    Detection Prevalence : 0.2349
##       Balanced Accuracy : 0.9508
##
##        'Positive' Class : Cancer
##
```

Model 3 statistics:

```
## Random Forest
##
## 5487 samples
##  303 predictor
##    2 classes: 'Cancer', 'Non-Cancer'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 4939, 4938, 4938, 4939, 4938, 4938, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##     2   0.8997014  0.6862133
##    52   0.9684683  0.9133157
##   102   0.9673138  0.9105090
##   152   0.9669492  0.9094471
##   202   0.9658555  0.9064951
##   252   0.9650653  0.9044503
##   303   0.9631821  0.8993428
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 52.
```

The predictive results of this third model are once again similar to model 1 results and this was expected. In fact, we already saw (**Figure 1**) that there was no specific text length distribution between "cancer" and "non cancer" trials.

3.5 Model 4: 300 v features from bigrams

Here we repeated the model 1 simulation but this we add word order importance to our new model by considering bigrams instead of unigrams.

Model 4 confusion matrix:

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Cancer Non-Cancer
##   Cancer        481         19
##   Non-Cancer     96       1754
##
##                Accuracy : 0.9511
##                  95% CI : (0.9416, 0.9594)
##     No Information Rate : 0.7545
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8617
##
##  Mcnemar's Test P-Value : 1.37e-12
##
##             Sensitivity : 0.8336
##             Specificity : 0.9893
##          Pos Pred Value : 0.9620
##          Neg Pred Value : 0.9481
##              Prevalence : 0.2455
##          Detection Rate : 0.2047
##    Detection Prevalence : 0.2128
##       Balanced Accuracy : 0.9115
##
##        'Positive' Class : Cancer
##
```

Model 4 statistics:

```
## Random Forest
##
## 5487 samples
##  300 predictor
##    2 classes: 'Cancer', 'Non-Cancer'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 4939, 4938, 4938, 4939, 4938, 4938, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##     2   0.9371819  0.8153118
##    51   0.9576541  0.8816988
##   101   0.9563791  0.8781337
##   151   0.9551032  0.8746094
##   200   0.9537671  0.8707796
##   250   0.9526732  0.8678368
##   300   0.9517009  0.8648678
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 51.
```

3.6 Model 5: 300 v features from unigrams and bigrams

The previous model is based only on bi-grams and was little bit less effective than the models based on unigrams (models 1 to 3). In this predictive model n°5 we combined unigrams and bigrams

PS: due to the extreme high number of tokens (88 554 !), we had to trim our DFM by removing tokens occurring less than 2 times overall or in less than 2 documents (trials).

Model 5 confusion matrix:

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Cancer Non-Cancer
##   Cancer        522         27
##   Non-Cancer     55       1746
##
##                Accuracy : 0.9651
##                  95% CI : (0.9569, 0.9722)
##     No Information Rate : 0.7545
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9043
##
##  Mcnemar's Test P-Value : 0.002867
##
##             Sensitivity : 0.9047
##             Specificity : 0.9848
##          Pos Pred Value : 0.9508
##          Neg Pred Value : 0.9695
##              Prevalence : 0.2455
##          Detection Rate : 0.2221
##    Detection Prevalence : 0.2336
##       Balanced Accuracy : 0.9447
##
##        'Positive' Class : Cancer
##
```

Model 5 statistics:

```
## Random Forest
##
## 5487 samples
##  300 predictor
##    2 classes: 'Cancer', 'Non-Cancer'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 4939, 4938, 4938, 4939, 4938, 4938, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##     2   0.9257610  0.7767312
##    51   0.9690765  0.9149498
##   101   0.9683464  0.9131988
##   151   0.9667059  0.9088317
##   200   0.9657948  0.9064783
##   250   0.9654908  0.9056160
##   300   0.9643974  0.9026406
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 51.
```

4 DISCUSSION

Here is the summary of the Random Forest algorithm results of all the models tested in this report :

Predictive Model #	Accuracy	Sensitivity	Specificity	Optimal RF mtry	Computation time*
Model 1: 300 v features from unigrams	0.9685	0.9133	0.9865	51	7h 15min
Model 2: 500 v features from unigrams	0.9681	0.9151	0.9853	79	11h 40min
Model 3: 300 v features from unigrams + 3 text length features	0.9689	0.9151	0.9865	52	7h 55min
Model 4: 300 v features from bigrams	0.9511	0.8336	0.9893	51	7h 10min
Model 5: 300 v features from unigrams and bigrams	0.9651	0.9047	0.9848	51	7h 05min

* *using 3 cores processor.*

The selected predictive model for our Random Forest manually labelled data in a tenfold cross-validation repeated 3 times is **Model 1**. This model has an ***accuracy*** of ***96.86%***, a ***sensitivity*** of ***91.33%***, and a ***specificity*** of ***98.65%***. These results are very closed to Model 3 results and with ***less computing time***.

This sensitivity would usually not suffice, however, since the cross-validation was performed on data that was selected on the basis of high similarities, that data contained many observations that were relatively difficult to classify.

This clinical trials automatic cancer labeling system can be further optimized by considering Machine Learning upgrades. We used the Random Forest algorithm and the reason is that it is relatively simple and powerful. There are other models that can be used like **Support Vector Machine** (**SVM**) or **Boost Decision Trees**.

The field of application of this automatic labelling can be extended by applying it not only to determine if a trial is related to cancer but also to determine the **cancer category** (solid/liquid or by organ).

APPENDICES

Packages used in this report

```
##         xml2      forcats        dplyr        purrr        readr       tibble
##      "1.3.2"      "0.5.1"      "1.0.7"      "0.3.4"      "2.0.0"      "3.1.2"
##    tidyverse     tidytext        tidyr      stringr        rvest       readxl
##      "1.3.1"      "0.3.1"      "1.1.3"      "1.4.0"      "1.0.0"      "1.3.1"
## randomForest     quanteda     openxlsx          lsa    SnowballC    lubridate
##     "4.6-14"      "3.1.0"      "4.2.4"    "0.73.2"      "0.7.0"     "1.7.10"
##   kableExtra         irlba       Matrix         httr    gridExtra      ggrepel
##      "1.3.4"      "2.3.5"    "1.2-18"      "1.4.2"        "2.3"      "0.9.1"
##        e1071       doSNOW         snow    iterators      foreach   data.table
##      "1.7-9"     "1.0.19"      "0.4-4"     "1.0.13"      "1.5.1"     "1.14.0"
##        caret      lattice      ggplot2
##     "6.0-90"    "0.20-41"      "3.3.5"
```

Codes ran for this report

```
knitr::opts_chunk$set(echo = TRUE, fig.pos = 'H')
# Packages installation

if(!require(caret))
  install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table))
  install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(doSNOW))
  install.packages("doSNOW", repos = "http://cran.us.r-project.org")
if(!require(e1071))
  install.packages("e1071", repos = "http://cran.us.r-project.org")
if(!require(ggplot2))
  install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(ggrepel))
  install.packages("ggrepel", repos = "http://cran.us.r-project.org")
if(!require(gridExtra))
  install.packages("gridExtra", repos = "http://cran.us.r-project.org")
if(!require(httr))
  install.packages("httr", repos = "http://cran.us.r-project.org")
if(!require(irlba))
  install.packages("irlba", repos = "http://cran.us.r-project.org")
if(!require(kableExtra))
  install.packages("kableExtra", repos = "http://cran.us.r-project.org")
if(!require(lubridate))
  install.packages("lubridate", repos = "http://cran.us.r-project.org")
if(!require(lsa))
  install.packages("lsa", repos = "http://cran.us.r-project.org")
if(!require(openxlsx))
  install.packages("openxlsx", repos = "http://cran.us.r-project.org")
if(!require(quanteda))
  install.packages("quanteda", repos = "http://cran.us.r-project.org")
if(!require(randomForest))
  install.packages("randomForest", repos = "http://cran.us.r-project.org")
if(!require(readxl))
  install.packages("readxl", repos = "http://cran.us.r-project.org")
if(!require(rvest))
  install.packages("rvest", repos = "http://cran.us.r-project.org")
if(!require(SnowballC))
  install.packages("SnowballC", repos = "http://cran.us.r-project.org")
if(!require(stringr))
  install.packages("stringr", repos = "http://cran.us.r-project.org")
if(!require(tidyr))
  install.packages("tidyr", repos = "http://cran.us.r-project.org")
if(!require(tidytext))
  install.packages("tidytext", repos = "http://cran.us.r-project.org")
if(!require(tidyverse))
  install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(xml2))
  install.packages("tidyverse", repos = "http://cran.us.r-project.org")

# Packages loading
```

```r
library(caret)
library(data.table)
library(doSNOW)
library(e1071)
library(ggplot2)
library(ggrepel)
library(gridExtra)
library(httr)
library(irlba)
library(kableExtra)
library(lubridate)
library(lsa)
library(openxlsx)
library(quanteda)
library(randomForest)
library(readxl)
library(rvest)
library(SnowballC)
library(stringr)
library(tidyr)
library(tidytext)
library(tidyverse)
library(xml2)

# Get from my github repository the Xml file downloaded from ICTRP (12 586 trials)
temp <- tempfile()
download.file(
  "https://github.com/vakiki/Clinical-Trials-Project/raw/main/ICTRP-Results_mi2018.zip"
  ,temp)
dlfile <- read_xml(unz(temp, "ICTRP-Results_mi2018.xml"))
unlink(temp)

# Select the wanted items

ICTRP_TrialID <- xml_find_all(dlfile,"//Internal_Number")
ICTRP_TrialID <- xml_text(ICTRP_TrialID, trim = TRUE)
ICTRP_TrialID <- as.data.frame(ICTRP_TrialID)

Original_TrialID <- xml_find_all(dlfile,"//TrialID")
Original_TrialID <- xml_text(Original_TrialID, trim = TRUE)
Original_TrialID <- as.data.frame(Original_TrialID)

Public_title <- xml_find_all(dlfile,"//Public_title")
Public_title <- xml_text(Public_title, trim = TRUE)
Public_title <- as.data.frame(Public_title)

Scientific_title <- xml_find_all(dlfile,"//Scientific_title")
Scientific_title <- xml_text(Scientific_title, trim = TRUE)
Scientific_title <- as.data.frame(Scientific_title)

Study_Source <- xml_find_all(dlfile,"//Source_Register")
Study_Source <- xml_text(Study_Source, trim = TRUE)
Study_Source <- as.data.frame(Study_Source)
```

```r
Study_type <- xml_find_all(dlfile,"//Study_type")
Study_type <- xml_text(Study_type, trim = TRUE)
Study_type <- as.data.frame(Study_type)

Study_Condition <- xml_find_all(dlfile,"//Condition")
Study_Condition <- xml_text(Study_Condition, trim = TRUE)
Study_Condition <- as.data.frame(Study_Condition)


# Merge all the previous variable into one table

ICTRP_table <- as.data.frame(c(ICTRP_TrialID,
                               Original_TrialID,
                               Study_Source,
                               Public_title,
                               Scientific_title,
                               Study_Condition,
                               Study_type))


# Remove all unneeded objects
rm(list = c("dlfile",
            "ICTRP_TrialID",
            "Original_TrialID",
            "Study_Source",
            "Public_title",
            "Scientific_title",
            "Study_Condition",
            "Study_type"))

# Total number of records within the original dataset
N_before <- nrow(ICTRP_table)

# Check for missing values in our dataset
length(which(!complete.cases(ICTRP_table)))

# Delete duplicates within clinical trials records

# 1/ Based on the "Original trial ID"
ICTRP_table <- distinct(ICTRP_table, Original_TrialID, .keep_all = TRUE)
# 2/ Based on the "Scientific title" text
ICTRP_table <- distinct(ICTRP_table, Scientific_title, .keep_all = TRUE)
# 3/ Based on the "Public title" text
ICTRP_table <- distinct(ICTRP_table, Public_title, .keep_all = TRUE)

# Total number of records after deleting the previous duplicates
N_After <- nrow(ICTRP_table)

# Number and percentage of duplicates in the original dataset
Delta = N_before - N_After
perc = (Delta/N_before) * 100

# Dataset "Study Type" categories distribution
```

```r
StudyTypeCategories_before <- ICTRP_table %>% group_by(Study_type) %>% count()

StudyTypeCategories_before %>%
  kable("latex",booktabs=TRUE,caption ="Study Type initial categories distribution")%>%
  kable_styling(latex_options = c("HOLD_position", "striped"))

#  "Study Type" categories standardization : uppercase and lowercase
ICTRP_table$Study_type <- str_to_title(ICTRP_table$Study_type)

# Dataset "Study Type" categories distribution after standardization
StudyTypeCategories_after <- ICTRP_table %>% group_by(Study_type) %>% count()

# Create an object containing all the Clinical trials (= Interventional)
Interventional = c("Intervention",
                   "Interventional",
                   "Interventional Clinical Trial Of Medicinal Product",
                   "Interventional Study")

# Get only the clinical trials (= Interventional) from our dataset
ICTRP_table <- ICTRP_table %>% filter(Study_type %in% Interventional)

# Number of records left after duplicates deletion and Interventional Study selection
N_Interventional <- nrow(ICTRP_table)

# Call the Excel file with the "Cancer / Non-Cancer" labels for studies
xlURL <-
"https://github.com/vakiki/Clinical-Trials-Project/raw/main/Cancer_Not_Cancer_table.xlsx"
GET(xlURL, write_disk(temp <- tempfile(fileext = ".xlsx")))
CancerLabels_ReferenceTable <- read_excel(temp, sheet = "List")
unlink(temp)

# Keep only the "Trial ICTRP ID" and the label "Cancer/Non-Cancer"
CancerLabels_ReferenceTable <- CancerLabels_ReferenceTable %>%
                               select(Cancer,ICTRP_TrialID)

# Integrate the "Cancer / Non-Cancer" labels to our dataset
ICTRP_table <- inner_join(ICTRP_table,CancerLabels_ReferenceTable,by = "ICTRP_TrialID")

# Convert the class label into a factor (useful for ggplot2 per example)
ICTRP_table$Cancer <- as.factor(ICTRP_table$Cancer)
ICTRP_table %>% group_by(Cancer) %>% count()

# Cancer labels distribution within the dataset
CancerLabelsDistribution <- round(prop.table(table(ICTRP_table$Cancer)),3)

CancerLabelsDistribution %>%
  kable("latex",booktabs=TRUE,caption = "Cancer/Non-Cancer label distribution")%>%
  kable_styling(latex_options = c("HOLD_position", "striped"))

# Only keep the necessary column for the upcoming text analysis process
ICTRP_table <- ICTRP_table %>%
              select(ICTRP_TrialID,Public_title,Scientific_title,Study_Condition,Cancer)
```

```r
# Analyze the text lengths of "Public_title", "Scientific_title", "Study_Condition"
# Create 3 columns for text length of "Public_title", "Scientific_title", "Study_Condition"
ICTRP_table$Public_title_TextLength <- nchar(ICTRP_table$Public_title)
ICTRP_table$Scientific_title_TextLength <- nchar(ICTRP_table$Scientific_title)
ICTRP_table$Study_Condition_TextLength <- nchar(ICTRP_table$Study_Condition)

# Distribution of "Public_title" text length with Class Labels
Public_title_textlength <- ggplot(ICTRP_table,
                                  aes(x = Public_title_TextLength,
                                      fill = Cancer)) +
  theme_bw() +
  geom_histogram(binwidth = 5) +
  labs(y = "Text Count", x = "Length of Text",
       title = "Distribution of Public_title text length with Class Labels")

# Distribution of "Scientific_title" text length with Class Labels
Scientific_title_textlength <- ggplot(ICTRP_table,
                                      aes(x = Scientific_title_TextLength,
                                          fill = Cancer)) +
  theme_bw() +
  geom_histogram(binwidth = 5) +
  labs(y = "Text Count", x = "Length of Text",
       title = "Distribution of Scientific_title text length with Class Labels")

# Distribution of "Study_Condition" text length with Class Labels
Study_condition_textlength <- ggplot(ICTRP_table,
                                     aes(x = Study_Condition_TextLength,
                                         fill = Cancer)) +
  theme_bw() +
  geom_histogram(binwidth = 5) +
  labs(y = "Text Count", x = "Length of Text",
       title = "Distribution of Study_Condition text length with Class Labels")

grid.arrange(Public_title_textlength,
             Scientific_title_textlength,
             Study_condition_textlength,
             nrow = 3)

# Concatenate "Public_title", "Scientific_title", "Study_Condition" into "text"
ICTRP_table <- ICTRP_table %>% unite("Text",
                                     Public_title:Study_Condition,
                                     sep= " ",
                                     na.rm = TRUE,
                                     remove = TRUE)

# Remove all unnecessary columns for the data analysis process
ICTRP_table <- ICTRP_table %>% select(ICTRP_TrialID,
                                      Text,Public_title_TextLength,
                                      Scientific_title_TextLength,
                                      Study_Condition_TextLength,
                                      Cancer)

# Rename columns of our dataset
```

```r
names(ICTRP_table) <- c("ID",
                        "Text",
                        "Pub_TextLength",
                        "Sci_TextLength",
                        "Cond_TextLength",
                        "Label")

# Convert our class label into a factor
ICTRP_table$Label <- as.factor(ICTRP_table$Label)

#STEP 1 : CREATE A 70%/30% STRATIFIED SPLIT

# Seting the random seed for reproducibility
set.seed(21)

# Create a 70% /30% stratified split dataset
indexes <- createDataPartition(ICTRP_table$Label, times = 1, p =0.7, list = FALSE)

# Identify our train and test datasets
train <- ICTRP_table[indexes,]
test <- ICTRP_table[-indexes,]

# Cancer labels distribution within the dataset
CancerLabelsTrainDistribution <- round(prop.table(table(train$Label)),4)

CancerLabelsTrainDistribution %>%
  kable("latex", booktabs=TRUE, caption = "Train Cancer/Non Cancer distribution") %>%
  kable_styling(latex_options = c("HOLD_position", "striped"))

# Cancer labels distribution within the dataset
CancerLabelsTestDistribution <- round(prop.table(table(test$Label)),4)

CancerLabelsTestDistribution %>%
  kable("latex", booktabs=TRUE, caption = "Test Cancer/Non Cancer distribution") %>%
  kable_styling(latex_options = c("HOLD_position", "striped"))

# Tokenize our train text and remove digits ; punctuations ; symbols
train.tokens <- tokens(train$Text, what = "word",
                       remove_numbers = TRUE, remove_punct = TRUE,
                       remove_symbols = TRUE)

# Lower case the train tokens
train.tokens <- tokens_tolower(train.tokens)

# Remove "stop words" using quanteda's built-in English stopword list
train.tokens <- tokens_select(train.tokens, stopwords(),
                              selection = "remove")
# Perform stemming on the train tokens
train.tokens <- tokens_wordstem(train.tokens, language = "english")

# Add bigrams (only for model 4 & 5)
# train.tokens <- tokens_ngrams(train.tokens, n = 1:2)
```

```r
# Transform our train tokens into a DFM : the "bag of words" model
train.tokens.DFM <- dfm(train.tokens, tolower = FALSE)

# Transform our train DFM into matrix format (for data exploration purpose)
train.tokens.DFM_matrix <- as.matrix(train.tokens.DFM)

# Setup the feature data frame with labels
train.tokens.DFM.labels_df <- cbind(Label = train$Label, data.frame(train.tokens.DFM))

# Cleanup column names
names(train.tokens.DFM.labels_df) <- make.names(names(train.tokens.DFM.labels_df))

# Calculate the TF-IDF of our train DFM
train.tokens.DFM.TFIDF <- dfm_tfidf(x = train.tokens.DFM,
                                    scheme_tf = "count",
                                    scheme_df = "inverse",
                                    base = 10,
                                    force = FALSE)

# Transform the train TF-IDF into a matrix format
train.tokens.DFM.TFIDF_matrix <- as.matrix(train.tokens.DFM.TFIDF)

# Check for incomplete cases
incomplete.cases <- which(!complete.cases(train.tokens.DFM.TFIDF_matrix))

# Fix incomplete cases
train.tokens.DFM.TFIDF_matrix[incomplete.cases,] <- rep(0.0,
                                                        ncol(train.tokens.DFM.TFIDF_matrix))

# Make a clean data frame using the same process as before
train.tokens.DFM.TFIDF.labels_df <- cbind(Label = train$Label,
                                          data.frame(train.tokens.DFM.TFIDF))
names(train.tokens.DFM.TFIDF.labels_df) <- make.names(
                                            names(train.tokens.DFM.TFIDF.labels_df))

# You can load the model 1 train SVD without running again the SVD simulation :
M1_train.svd <- readRDS(gzcon(url(
  "https://github.com/vakiki/Clinical-Trials-Project/raw/main/M1_train.svd")))

# Tokenize our test text and remove digits ; punctuations ; symbols
test.tokens <- tokens(test$Text, what = "word",
                      remove_numbers = TRUE, remove_punct = TRUE,
                      remove_symbols = TRUE)

# Lower case the test tokens
test.tokens <- tokens_tolower(test.tokens)

# Stopword removal from the test tokens
test.tokens <- tokens_select(test.tokens, stopwords(),
                             selection = "remove")

# Stemming the test tokens
test.tokens <- tokens_wordstem(test.tokens, language = "english")
```

```r
# Add bigrams (only for model 4 & 5)
# test.tokens <- tokens_ngrams(test.tokens, n = 1:2)

# Convert n-grams to document-term frequency matrix.
test.tokens.DFM <- dfm(test.tokens, tolower = FALSE)

# Apply train DFM n-grams to test DFM to have the same features in both datasets.
test.tokens.DFM <- dfm_match(test.tokens.DFM, features = featnames(train.tokens.DFM))
test.tokens.DFM_matrix <- as.matrix(test.tokens.DFM)

# Calculate the TF-IDF of our test DFM that has now the same features as the train DFM
test.tokens.DFM.TFIDF <- dfm_tfidf(x = test.tokens.DFM,
                                   scheme_tf = "count",
                                   scheme_df = "inverse",
                                   base = 10,
                                   force = FALSE)

# Transform the test TF-IDF into a matrix format
test.tokens.DFM.TFIDF_matrix <- as.matrix(test.tokens.DFM.TFIDF)

# Fix incomplete cases
test.tokens.DFM.TFIDF_matrix[is.na(test.tokens.DFM.TFIDF_matrix)] <- 0.0

# Load the model without running again the machine learning simulation :
M1_rf.cv.unigram.nv300.nolength <- readRDS(gzcon(url(
"https://github.com/vakiki/Clinical-Trials-Project/raw/main/M1_rf.cv.unigram.nv300.nolength.rds")))
M1_test.svd <- readRDS(gzcon(url(
"https://github.com/vakiki/Clinical-Trials-Project/raw/main/M1_test.svd.rds")))

# Model 1 confusion matrix
preds <- predict(M1_rf.cv.unigram.nv300.nolength, M1_test.svd)
confusionMatrix(preds, M1_test.svd$Label)

# Model 1 statistics
M1_rf.cv.unigram.nv300.nolength

# Load the model without running again the machine learning simulation :
M2_rf.cv.unigram.nv500.nolength <- readRDS(gzcon(url(
"https://github.com/vakiki/Clinical-Trials-Project/raw/main/M2_rf.cv.unigram.nv500.nolength.rds")))
M2_test.svd <- readRDS(gzcon(url(
"https://github.com/vakiki/Clinical-Trials-Project/raw/main/M2_test.svd.rds")))

# Model 2 confusion matrix
preds <- predict(M2_rf.cv.unigram.nv500.nolength, M2_test.svd)
confusionMatrix(preds, M2_test.svd$Label)

# Model 2 statistics
M2_rf.cv.unigram.nv500.nolength

# Load the model without running again the machine learning simulation :
M3_rf.cv.unigram.nv300.withlength <- readRDS(gzcon(url(
"https://github.com/vakiki/Clinical-Trials-Project/raw/main/M3_rf.cv.unigram.nv300.withlength.rds")))
M3_test.svd <- readRDS(gzcon(url(
```

```
"https://github.com/vakiki/Clinical-Trials-Project/raw/main/M3_test.svd.rds")))

# Model 3 confusion matrix
preds <- predict(M3_rf.cv.unigram.nv300.withlength, M3_test.svd)
confusionMatrix(preds, M3_test.svd$Label)

# Model 3 statistics
M3_rf.cv.unigram.nv300.withlength

# Load the model without running again the machine learning simulation :
M4_rf.cv.bigram.nv300.nolength <- readRDS(gzcon(url(
"https://github.com/vakiki/Clinical-Trials-Project/raw/main/M4_rf.cv.bigram.nv300.nolength.rds")))
M4_test.svd <- readRDS(gzcon(url(
"https://github.com/vakiki/Clinical-Trials-Project/raw/main/M4_test.svd.rds")))

# Model 4 confusion matrix
preds <- predict(M4_rf.cv.bigram.nv300.nolength, M4_test.svd)
confusionMatrix(preds, M4_test.svd$Label)

# Model 4 statistics
M4_rf.cv.bigram.nv300.nolength

# Load the model without running again the machine learning simulation :
M5_rf.cv.unigram.bigram.nv300.nolength <- readRDS(gzcon(url(
"https://github.com/vakiki/Clinical-Trials-Project/raw/main/M5_rf.cv.unigram.bigram.nv300.nolength.rds")
M5_test.svd <- readRDS(gzcon(url(
"https://github.com/vakiki/Clinical-Trials-Project/raw/main/M5_test.svd.rds")))

# Model 5 confusion matrix
preds <- predict(M5_rf.cv.unigram.bigram.nv300.nolength, M5_test.svd)
confusionMatrix(preds, M5_test.svd$Label)

# Model 5 statistics
M5_rf.cv.unigram.bigram.nv300.nolength


# Packages used and their version.
installed.packages()[names(sessionInfo()$otherPkgs), "Version"]
```