Zaawansowane Uczenie Maszynowe
Projekt analityczny
Dokumentacja końcowa

G. Artur, W. Wojciech

Czerwiec 2021

1 Cel projektu

Celem projektu było wykonanie oraz zbadanie klasyfikatora, który na podstawie szeregu atrybutów oszacowywał prawdopodobieństwo zmiany pracy wśród pracowników. Przy użyciu narzędzi języka R została przeprowadzona wnikliwa analiza danych oraz zbudowano modele z wykorzystaniem wybranych algorytmów. Zakres prac, które wykonano w ramach zadania obejmowały:

- przygotowanie zbioru danych (m.in. transformacja atrybutów, normalizacja, balansowanie klas)
- podstawowy statystyczny opis danych
- strojenie parametrów algorytmów
- utworzenie modeli wraz z oceną ich jakości
- analiza otrzymanych wyników

2 Zbiór danych

Dane te zostały zebrane przez firmę działającą w sektorze Analizy Danych i Big Data wśród osób, które uczęszczały na kursy prowadzone przez ów firmę. Cel jaki przyświecał zbieraniu danych to określenie, którzy z kursantów będą chętni zmienić pracę czy też podjąć się pracy w firmie, w przypadku, kiedy nie są jeszcze zatrudnieni. Jest to szczególnie istotne, ponieważ pozwala to na redukcję kosztu i czasu poświęcanego na wyszukiwanie kandydatów oraz zwiększenie jakości samych kursów i ich planowania. Zbiór danych pochodzi z konkursu na serwisie *Kaggle*[4]. W związku z tym, iż jest to konkurs na Kaggle-u autorzy dostarczyli również zbiór testujący, a co za tym idzie nie ma potrzeby wydzielania odrębnego zbioru testującego z macierzystego zbioru danych. Dostarczony zbiór trenujący składał się z 19 158 obserwacji a testujący z 2129 obserwacji.

2.1 Opis danych

W tablicy 1 przedstawiono listę atrybutów, która znajduję się w badanym zbiorze danych. Atrybut *target* był atrybutem docelowym zadania klasyfikacji. Dokładniejszy (statystyczny) opis atrybutów numerycznych został przedstawiony w tablicy 2 (atrybut *enrollee_id* został pominięty, ze względu na to, iż była to wartość unikalna dla każdego kursanta, co bezpośrednio przekładało się na unikalną wartości tego atrybutu dla każdej obserwacji). Adekwatny opis atrybutów kategorycznych przedstawiono w tablicy 3. Z punktu widzenia poprawności trenowania klasyfikatora na danym zbiorze danych, zbiór ten posiadał kilka cech, na które należy zwrócić szczególną uwagę, co zresztą zostało podkreślone na stronie konkursu. Wspomniane cechy to:

- zbiór był niezbalansowany,
- większość atrybutów była typu kategorycznego,
- istniały atrybuty ze znaczną ilością brakujących wartości.

Atrybut	**Typ**	**Opis**
enrollee_id	numeryczny	Identyfikator kursanta
city	kategoryczny	Kod miasta
city_development_index	numeryczny	Wskaźnik rozwoju miasta
gender	kategoryczny	Płeć
relevent_experience	kategoryczny	Istotne doświadczenie kursanta
enrolled_university	kategoryczny	Rodzaj studiów
education_level	kategoryczny	Wykształcenie
major_discipline	kategoryczny	Główny zawód
experience	kategoryczny	Doświadczenie zawodowe w latach
company_size	kategoryczny	Liczba zatrudnionych w miejscu pracy
company_type	kategoryczny	Rodzaj aktualnego pracodawcy
last_new_job	kategoryczny	Okres od ostatniej zmiany pracy w latach
training_hours	numeryczny	Liczba ukończonych godzin w kursach
target	kategoryczny	Chęć zmiany pracy

Tablica 1: Opis atrybutów

Atrybut	**Średnia**	**Odchylenie standardowe**	**Wartość minimalna**	**Wartość maksymalna**
training_hours	65,4	60,1	1	336
city_development_index	0,83	0,12	0,45	0,95

Tablica 2: Opis atrybutów numerycznych (zbiór treningowy)

2.2 Transformacja atrybutów kategorycznych

Znaczna część atrybutów w zbiorze danych była typu kategorycznego, w przypadku gdy algorytm klasyfikujący wymagał na wejściu atrybutów w postaci numerycznej przekształcono je do wymaganej reprezentacji. W celu zamiany atrybutów kategorycznych (nominalnych) na numeryczne zastosowano kodowanie typu *Dummy variable* przy pomocy metody *dummyVars*[2] z pakietu *caret*. W przypadku atrybutów kategorycznych (porządkowych) zastosowano standardowe podejście w postaci zamiany wartości kategorycznych na kolejne liczby naturalne na podstawie ustalonej hierarchii, w tym celu użyto metody *factor* z pakietu bazowego.

2.3 Eliminacja brakujących wartości

Jako podstawową metodę mającą na celu eliminacje brakujących wartości zastosowano imputację wartością średnią lub modalną.

Atrybut	**Liczba klas**	**Brakujące wartości**	**Najliczniejsza klasa**
city	123	0	city_103 (23%)
gender	3	4508 (24%)	Male (69%)
relevent_experience	2	0	Has relevent... (72%)
enrolled_university	3	386 (2%)	no_enrollment (72%)
education_level	5	460 (2%)	Graduate (61%)
major_discipline	6	2813 (15%)	STEM (76%)
experience	22	65 (∼0%)	>20 (17%)
company_size	8	5938 (31%)	50-99 (16%)
company_type	6	6140 (32%)	Pvt Ltd (51%)
last_new_job	6	423 (2%)	1 (42%)
target	2	0	0 (75%)

Tablica 3: Opis atrybutów kategorycznych (zbiór treningowy)

2.4 Normalizacja wartości

Normalizacja wartości sprowadziła się do odjęcia średniej arytmetycznej oraz podzielenia przez odchylenie standardowe atrybutów wymagających normalizacji. Jedyny atrybut wymagający (choć bardziej pasuje stwierdzeniem, iż normalizacja nic nie zaszkodzi) normalizacji to *training_hours*.

2.5 Wybór najistotniejszych atrybutów

W celu przeprowadzenia próby ewentualnego uproszczenia modelu wykorzystano selekcję najistotniejszych atrybutów metodą *Boruta* (z pakietu Boruta[1]).

2.6 Balansowanie klas

Problem niezbalansowanych klas występuje, kiedy w zbiorze treningowym rozkład wartości klasy docelowej znacząco odbiega od rozkładu jednostajnego, przy czym słowo „znacząco" jest przekładane na wartość procentową dosyć arbitralnie, tj. nie ma ściśle określonego rozłożenia, od którego możemy nazwać zbiór zbiorem niezbalansowanym. W zbiorze treningowym tyczącym się naszego zadania 75% ze wszystkich obserwacji należało do jednej klasy, nie był to ekstremalnie niezbalansowany przypadek, jednakże zastosowanie odpowiednich technik może zauważalnie poprawić skuteczność klasyfikacji. Przeprowadziliśmy eksperymentów na poniższych technikach, mianowicie:

- *Weight balancing* - przypisanie odpowiednich wag klasom podczas wyznaczania funkcji straty,
- próbkowanie przykładów klas często występujących (metoda *sample* z pakietu ROSE[9]),

3 Strojenie modeli

W celu dobrania odpowiednich parametrów wykorzystano metodę *k-krotnej walidacji krzyżowej* na zbiorze treningowym. Ogólna procedura wyglądała następująco:

1. Wymieszaj losowo zbiór
2. Podziel zbiór na k równoliczne podzbiory

3. Dla każdego podzbioru:
   (a) Wykorzystaj podzbiór jako zbiór walidacyjny
   (b) Wykorzystaj pozostałe podzbiory jako zbiór treningowy
   (c) Przeprowadź procedurę uczenia oraz walidacji
   (d) Zapisz wynik walidacji
4. Oblicz AUC

Należy podkreślić, że wszelkie ewentualne przekształcenia zbioru, opisane w poprzednich podrozdziałach, zostały zaaplikowane w pętli tylko na podzbiorach przeznaczonych do treningu w danej iteracji. Wartość parametru k została przyjęta jako 10, ponieważ jest to najczęściej występująca wartość w literaturze [7] oraz zazwyczaj modele charakteryzują się dobrymi wynikami. W celu stworzenia k-krotnej walidacji krzyżowej wykorzystano pakiet cvms [5]. Dla każdego algorytmu został stworzony indywidualny model, funkcja predykcyjna oraz ewaluacyjna. Dzięki dostosowaniu tego pakietu możliwe było równoczesne sprawdzenie wielu wariancji modelu (np. różnych formuł). Szczegółowe informacje o przetestowanych parametrach opisano w rozdziałach dotyczących poszczególnych algorytmów. Spośród przetestowanych rodzajów modeli wybrano ten wariant, który charakteryzował się największym AUC. Następnie przeprowadzono uczenie na całym zbiorze wykorzystując parametry tego modelu.

4 Procedura oceny modeli

Ocena jakości modeli została przeprowadzona z wykorzystaniem nieużywanego zbioru danych ewaluacyjnych, wykorzystując wyuczony najlepszy model. Następnie wyliczono współczynniki miar jakości klasyfikacji (*sensitivity*, *specificity*, *F1-score*), wykreślono krzywą ROC oraz obliczono prawdopodobieństwo AUC. Otrzymane wyniki porównano również z wynikami zamieszczonymi przez uczestników konkursu w serwisie Kaggle.

5 Implementacja

Implementacje naszego rozwiązania można podzielić na trzy człony:

- ładowanie i wstępne przetwarzanie danych
- k-krotna walidacja i trenowanie modeli,
- wybór najlepszego modelu i finalny trening.

Poddział ten jest odzwierciedlony w notatniku R-owym (*main.Rmd* w katalogu głównym), gdzie na samym początku mamy wspólną sekcje do ładowania danych oraz ich przetwarzania. następnie dla każdego z wybranych algorytmów mamy oddzielną sekcje, która wykorzystuje przygotowane dane do uczenia i strojenia modeli, a następnie dla następnego z nich możliwe jest przeprowadzenie uczenia na całym dostępnym zbiorze.

Główne wywołania tych metod znajdują się w notatniku który ma następującą strukturę:

1. Load data
2. Print base stats
3. Normalize data
4. Transform data
5. Create folds for cross validation

6. Train SVM model and predi
    (a) Results
        i. Balanced Accuracy, F1, MCC, Model ID, AUC
        ii. Best result
        iii. ROC
        iv. Train the best model on the whole training dataset
        v. Predict on test dataset
        vi. Plot confusion matrix
        vii. ROC
7. Train XGBoost model and predict
    (a) ...[1]
8. Train logistic regression model and predict
    (a) ...
9. Train Random Forest model and predict
    (a) ...

5.1 Ładowanie i wstępne przetwarzanie danych

Sekcja odnosi się do implementacji metod i technik przedstawionych w rozdziale 2. Przykładowo w notatniku fragment kodu przedstawiony poniżej odnosi się do przetworzenia wszystkich kategorycznych zmiennych, dla dostarczonych danych, do typu *factor* w języku R.

```
train_transformed <- train_normalized %>% transform_types(categorical_vars_names)
test_transformed <- test_normalized %>% transform_types(categorical_vars_names)
```

5.2 k-krotna walidacja i trenowanie modeli

W celu wykorzystania k-krotnej walidacji w naszym rozwiązani skorzystaliśmy z biblioteki *cvms*[5]. Biblioteka ta umożliwia zbudowanie elastycznego rozwiązania wykorzystującego różne modele o ile jesteśmy w stanie opakować funkcje uczącą i dokonującą predykcji w funkcje przejmujące i zwracające wartości określone przez autorów. Dodatkowo interfejs głównej funkcji wywołującej k-krotną walidację umożliwia przekazanie funkcji do pre-processingu (o ile spełnia odpowiednie wymagania co do argumentów i wartości zwracanej) oraz zestawy parametrów, dzięki czemu uzyskujemy kompletne rozwiązanie do strojenia modeli.

Przykładowo poniżej przedstawiono funkcję uczącą model oraz funkcję dokonującą predykcji dla algorytmu svm.

```
svm_model_fn <- function(train_data, formula, hyperparameters) {
  print("Training SVM - start")

  hyperparameters <- cvms::update_hyperparameters(
    kernel = "radial",
    cost = 1,
    use_rose = FALSE,
```

[1] Struktóra tak sama jak dla sekcji dotyczącej SVG

```
    use_weights = FALSE,
    hyperparameters = hyperparameters
  )

  # ROSE balancing
  if (hyperparameters[["use_rose"]]) {
    print("Training SVM - use ROSE")
    train_data <-ROSE(formula , data = train_data)$data
  }

  # Weight balancing.
  model_weights = NULL
  if (hyperparameters[["use_weights"]]) {
    print("Training lSVM - use weights")
    weights <- (train_data)
    model_weights = c("0"=weights$weight_0, "1"=weights$weight_1)
  }

  e1071::svm(formula = formula,
             data = train_data,
             type = 'C-classification',
             class.weights = model_weights,
             kernel = hyperparameters[["kernel"]],
             cost = hyperparameters[["cost"]],
             probability = TRUE)
}
```

```
model_predict <- function(test_data, model, formula, hyperparameters, train_data) {
  hyperparameters <- cvms::update_hyperparameters(
    dummy_model = NULL,
    hyperparameters = hyperparameters
  )
  # Check if dummy vars need to be applied
  if (!is.null(hyperparameters[["dummy_model"]])) {
    test_data <- test_data %>% add_dummy_vars(hyperparameters[["dummy_model"]])
  }

  predictions <- predict(object = model,
          newdata = test_data,
          allow.new.levels = TRUE,
          probability = TRUE)

  # Extract probabilities
  probabilities <- dplyr::as_tibble(attr(predictions, "probabilities"))

  # Return second column
  return(probabilities[[2]])
}
```

Mając do dyspozycji tak zaimplementowane funkcje dla każdego z wybranych modeli jesteśmy w stanie w wygodny sposób korzystać z dostrajania modeli. Wywołanie głównej funkcji odpowiadającej za k-krotną walidację przedstawiono poniżej.

```
cv_result <- cross_validate_fn(
  data = train_folded,
  formulas = c(model_formula),
  model_fn = svm_model_fn,
  predict_fn = model_predict,
  preprocess_fn = preprocess_fn,
  hyperparameters = list(
    "kernel" = c("linear", "radial"),
    "cost" = c(1, 2.5, 5),
    "use_rose" = c(TRUE)
  ),
  fold_cols = c(".folds"),
  type = "binomial",
  parallel = TRUE,
  verbose = TRUE
)
```

5.3 Wybór najlepszego modelu i finalny trening

Wyboru dokonujemy przez inspekcje obiektu zwróconego przez k-krotną walidacje. W takie inspekcji jesteśmy w stanie wybrać najlepszy model pod względem szeregu różnych metryk m.in. AUC. Po wyborze najlepszego modelu następuje wyciągnięcie wymaganych parametrów modelu tj. formuła i hiper-parametry z obiektu zwróconego przez k-krotną walidacje i wykorzystanie ów danych do wywołania metod zdefiniowanych w poprzednim podrozdziale, jednakże tym razem dla całego dostępnego zbioru danych. Fragment takiego procesu został przedstawiony poniżej.

```
best_model_id <- 2
best_result <- cv_result %>% dplyr::slice(best_model_id)
best_model_formula <- as.formula(paste(c(best_result$Dependent, best_result$Fixed),
↪  collapse = ' ~ '))
preprocessed_data <- preprocess_fn(train_transformed, test_transformed,
↪  best_model_formula, best_result$HParams[[1]])

best_model <- svm_model_fn(
  train_data = preprocessed_data$train,
  formula = best_model_formula,
  hyperparameters = best_result$HParams[[1]]
)
```

5.4 Instrukcja obsługi

W przypadku chęci uruchomienia rozwiązania należy pobrać kod z repozytorium `https://github.com/wwrzesien/hr-classifier`, a następnie otworzyć R-owy notatnik i podążać kolejnymi sekcjami (opisane wyżej) aż natrafimy na początek sekcji z trenowaniem, wtedy wybieramy jeden z algorytmów i podążamy krokami zawierającymi się w sekcji odnoszącej się do wybranego algorytmu.

Uruchomienie początkowych sekcji zainstaluje wszystkie wymagane biblioteki, przy czym należy zwrócić uwagę na stałą *JOB_CORES*, która określa liczbę logicznych rdzeni wykorzystywanych przy zrównoleglonych obliczaniach. Flagę tą należy ustawić na odpowiednią do naszych potrzeba wartość.

6 Algorytmy

W poniższych podrozdziałach przedstawiono analizę z wykorzystaniem 4 algorytmów: gradient boosting, SVM, las losowy i regresja logistyczna wraz z opisem przygotowania danych oraz uzyskanych wyników. Poniżej opisano etap przygotowania danych, który należy wykonać przed uruchomieniem modeli, oraz wybór atrybutów istotnych metodą Boruta.

Przygotowanie danych Po wczytaniu plików wejściowych oraz wyświetleniu podstawowych statystyk znormalizowano parametr *training_hours*. Następnie zmienne kategoryczne zostały przekonwertowane na typ *factor*, a liczbowe na *integer*. Po usunięciu kolumny *enrollee_id* przeprowadzono transformację atrybutów kategorycznych. Następnie zbiór treningowy został podzielony na 10 części.

Wybór istotnych atrybutów W celu uproszczenia modeli oraz ograniczenia ryzyka nadmiernego dopasowania została przeprowadzona selekcja atrybutów przy użyciu metody Boruta wykorzystującego las losowy. Spośród 12 dostępnych atrybutów (*enrollee_id* został usunięty, *target* jest atrybutem predykcji) algorytm sklasyfikował 7 jako istotne (*city, city_development_index, education_level, enrolled_university, experience, relevant_experience, last_new_job*), 4 jako nieistotne (*company_size, company_type, major_discipline, training_hours*) oraz 1 jako niepewny (*gender*). Na potrzeby analizy atrybut *gender* został dodany do atrybutów istotnych.

6.1 Las losowy

W celu przetestowania algorytmu lasu losowego wykorzystano pakiet randomForest [8]. W tabeli 4 przedstawiono oraz opisano zastosowane parametry algorytmu.

Parametr	**Wartość**	**Definicja**
ntree	10, 50, 100, 200	Liczba drzew w lasie losowym.
ROSE	True, False	Flaga wskazująca na użycie algorytmu ROSE w fazie preprocessingu.
mtry	Default, 100, 170	Liczba kandydatów losowanych w węźle. *Default* odnosi się do pierwiastka z ilości danych.
nodesize	1, 5, 10, 15	Minimalny rozmiar węzła terminalnego.
Imputacja	True	Flaga wskazująca za użycie imputacji średnią lub modalną w fazie preprocessingu

Tablica 4: Opis parametrów modelu dla lasu losowego

6.1.1 Wyniki dla wybranych konfiguracji parametrów

W przypadku k-krotnej walidacji krzyżowej dla lasu losowego wybór najlepszy parametrów został przeprowadzony wybiórczo tj. nie dla wszystkich kombinacji ze względu na ograniczenia wydajnościowe sprzętu oraz prawdopodobnie błędy w implementacji biblioteki odpowiadającej za k-krotną walidację. Mianowicie, dla bardziej złożonych modeli testy nigdy się nie kończyły pomimo tego, że można było zaobserwować brak obciążenia procesora procesu wykonującego testy. Było to szczególnie problematyczne ze względu na ewentualną jakość finalnego rozwiązania generowanego przez las losowy, ponieważ im większa liczba drzew do wygenerowania w lasie losowym tym lepsze on powinien dawać wyniki.

Poniższe tabele zawierają wyniki dla wybranych kombinacji parametrów. Zostało przedstawionych kilka tabeli z czego każda z nich odnosi się do innego zestawu atrybutów, ponieważ niektóre testowanie niektórych parametrów z innymi jednocześnie nie ma sensu. W tabelach przedstawiono maksymalnie 5 najlepszych pod względem AUC modeli.

ID	ntree	nodesize	mtry	ROSE	Imputacja	F1	MCC	Balanced Accuracy	AUC
4-2	100	5	DEFAULT	False	True	0.458	0.319	0.643	0.759
1-2	100	1	DEFAULT	True	True	0.544	0.372	0.701	0.757
3-2	100	1	DEFAULT	False	True	0.457	0.316	0.642	0.756
2-2	100	5	DEFAULT	True	True	0.540	0.365	0.698	0.755
4-1	10	5	DEFAULT	True	True	0.449	0.303	0.637	0.741

Tablica 5: Wyniki lasu losowego dla k-krotnej walidacji krzyżowej (zestaw 1)

Jak widać w tabeli 5 (i w kolejnych tabelach) las losowy osiąga sumarycznie gorsze wyniki jeśli stosujemy go z algorytmem ROSE. Jednakże, gorsze odnosi się tutaj do drobnej różnicy (często mniej niż jeden punkt procentowy). Dodatkowo można zauważyć, że parametry odpowiadające za większe uogólnianie modelu spisują się lepiej (zwłaszcza w wykorzystywanym zbiorze danych). Mowa tu o *mtry* i *nodesize*. W przypadku tego pierwszego im mniejsza jest jego wartość tym mniej bierzemy kandydatów pod uwagę podczas dokonywania decyzji podziału co prowadzi do tego, że klasy niedominujące mają większą szans na wpłynięcie na decyzję podziału. W przypadku zwiększenia drugiego wspomnianego parametru drzewa, które powstają w wyniku budowania modelu są mniejsze przez co ponownie lepiej generalizują. Oczywiście wszystko w granicach rozsądku, ponieważ zbytnie generalizowanie może przynieść skutek odwrotny do oczekiwanego co widać na przykładzie wielkości drzew, jeśli drzewa są za małe to jakość modelu spada (ponownie nie są to duże różnice).

ID	ntree	nodesize	mtry	ROSE	Imputacja	F1	MCC	Balanced Accuracy	AUC
11	50	10	DEFAULT	False	True	0.469	0.334	0.649	0.762
12	50	10	170	False	True	0.463	0.325	0.646	0.761
10	50	5	170	False	True	0.456	0.318	0.642	0.759
6	50	10	170	True	True	0.546	0.374	0.703	0.757
9	50	5	DEFAULT	False	True	0.467	0.328	0.648	0.757

Tablica 6: Wyniki lasu losowego dla k-krotnej walidacji krzyżowej (zestaw 2)

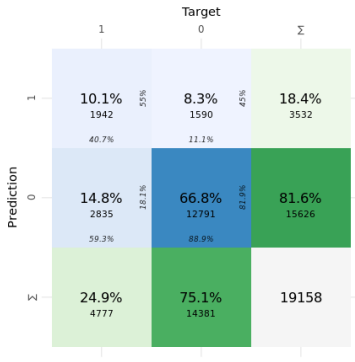
ID	ntree	nodesize	mtry	ROSE	Imputacja	F1	MCC	Balanced Accuracy	AUC
2	200	10	DEFAULT	False	True	0.470	0.334	0.650	0.765
1	200	10	DEFAULT	True	True	0.551	0.380	0.706	0.760

Tablica 7: Wyniki lasu losowego dla k-krotnej walidacji krzyżowej (zestaw 3)

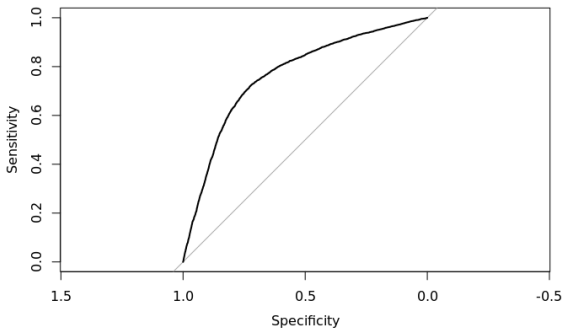
ID	ntree	nodesize	mtry	ROSE	Imputacja	F1	MCC	Balanced Accuracy	AUC
1	100	15	100	False	True	0.467	0.33	0.648	0.766

Tablica 8: Wyniki lasu losowego dla k-krotnej walidacji krzyżowej (zestaw 4)

Spośród wszystkich przetestowanych modeli najlepszą kombinacje parametrów osiągnięto dla modelu z tabeli 8 (na rysunku 1 przedstawiono macierz pomyłek i krzywą ROC). Dla tego modelu przeprowadzono ponowne uczenie na całym treningowym zbiorze danych po czym dokonano ewaluacji na zbiorze testowym. Metryki końcowego modelu przedstawiono w tabeli 9 natomiast wykresy znajdują się na rysunku 2. W tabeli 10 przedstawiono wyniki dla najlepszego modelu badanego na danych z ograniczoną liczbą kolumn metodą Boruta. Był to model o tych samych parametrach co dla pełnego zestawu cech, jednakże, jak można zauważyć, wyniki te są gorsze w porównaniu do modelu trenowanego na całym zbiorze danych ze wszystkimi kolumnami.



(a) Macierz pomyłek

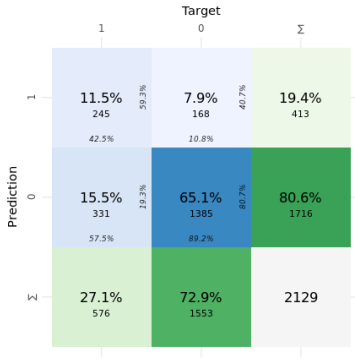


(b) Krzywa ROC

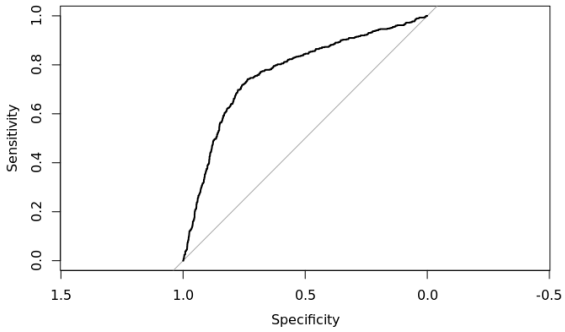
Rysunek 1: Las losowy dla k-krotnej walidacji krzyżowej

ID	ntree	nodesize	mtry	ROSE	Imputacja	F1	MCC	Balanced Accuracy	AUC
1	100	15	100	False	True	0.495	0.356	0.658	0.767

Tablica 9: Wyniki ewaluacji lasu losowego na zbiorze testowym



(a) Macierz pomyłek



(b) Krzywa ROC

Rysunek 2: Las losowy na zbiorze testowym

ID	ntree	nodesize	mtry	ROSE	Imputacja	F1	MCC	Balanced Accuracy	AUC
1	100	15	100	False	True	0.476	0.337	0.648	0.737

Tablica 10: Wyniki ewaluacji lasu losowego na zbiorze testowym z wykorzystaniem kolumn istotnych

6.2 Regresja logistyczna

W celu przetestowania algorytmu regresji logistycznej wykorzystano pakiet glmnet [3]. W tabeli 11 przedstawiono oraz opisano zastosowane parametry algorytmu. Z powodu problemów z implementacją wag klas w tym modelu tą analizę pominięto.

Parametr	**Wartość**	**Definicja**
Family	binomial	Zmienna definiująca binarną regresję logistyczną
ROSE	True, False	Flaga wskazująca na użycie algorytmu ROSE w fazie preprocessingu.
Imputacja	True	Flaga wskazująca za użycie imputacji średnią lub modalną w fazie preprocessingu

Tablica 11: Opis parametrów modelu

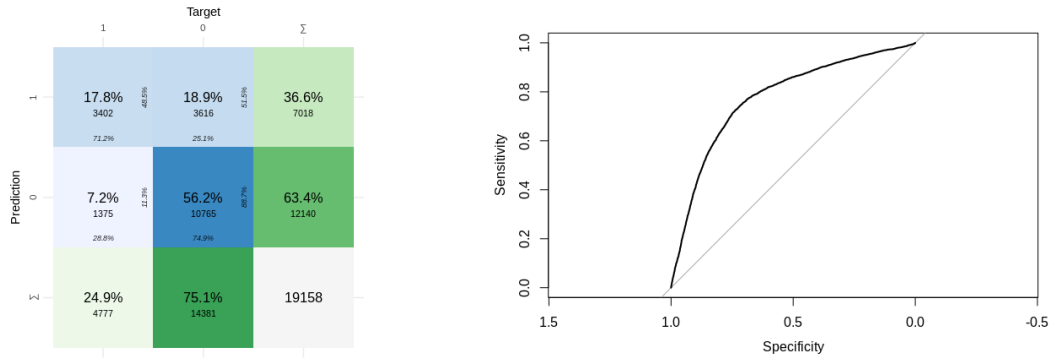
6.2.1 Wyniki dla wybranych konfiguracji parametrów

Podczas k-krotnej walidacji z wykorzystaniem wszystkich atrybutów otrzymano modele przedstawione w tabeli 12 wraz z wynikami poszczególnych metryk.

ID	**Family**	**ROSE**	**Imputacja**	**F1**	**MCC**	**Balanced Accuracy**	**AUC**
1	binomial	False	True	0.42	0.32	0.63	0.78
2	binomial	True	True	0.57	0.41	0.73	0.78

Tablica 12: Wyniki k-krotnej walidacji krzyżowej

Najlepszym modelem okazał się model nr. 2, w którym wykorzystano balansowanie klas metodą ROSE. Skuteczność obu modeli wyniosła 0.78 lecz drugi model osiągnął wyższe wartości pozostałych parametrów. Następnie dla tego modelu obliczono macierz pomyłek (rys. 3a) oraz wykreślono krzywą ROC (rys. 3b).



(a) Macierz pomyłek

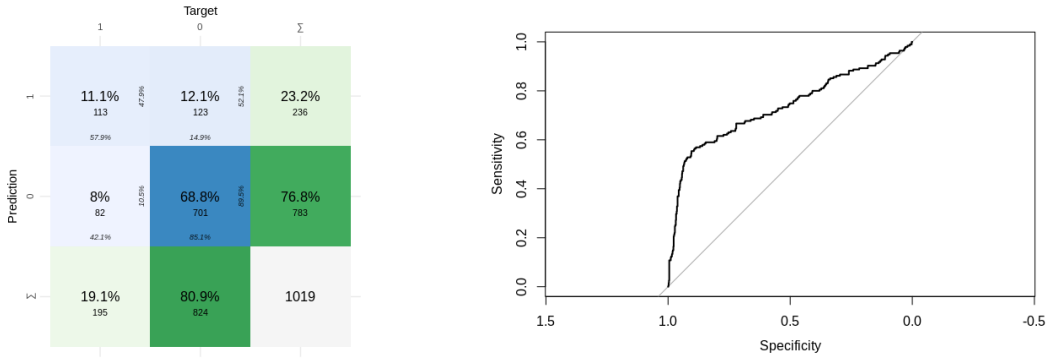
(b) Krzywa ROC

Rysunek 3: Regresja logistyczna dla k-krotnej walidacji krzyżowej

Następnie został on wytrenowany na pełnym zbiorze treningowym oraz przetestowany na zbiorze ewaluacyjnym. W tabeli 13 przedstawiono otrzymane wyniki. Macierz pomyłek oraz krzywą ROC zaprezentowano na rysunku 4a oraz 4b.

ID	Sensitivity	Specificity	F1	MCC	Balanced Accuracy	AUC
2	0.579	0.85	0.524	0.401	0.715	0.734

Tablica 13: Regresja logistyczna na zbiorze testowym



(a) Macierz pomyłek

(b) Krzywa ROC

Rysunek 4: Wyniki modelu na zbiorze ewaluacyjnym

Model na zbiorze ewaluacyjnym osiągnął gorsze rezultaty od modelu z walidacji krzyżowej. Różnica była niewielka i wyniosła 0.05 pkt. procentowego dla AUC oraz 0.046 dla F1.

W celu zbadania poprawy modelu, przeprowadzono kolejne uczenie na pełnym zbiorze tym razem wykorzystując atrybuty skategoryzowane przez algorytm Boruta za istotne. Analizując tabelę 14 można zauważyć poprawę w porównaniu do poprzedniego modelu w prawdopodobieństwie AUC oraz współczynniku F1. W pozostałych przypadkach uzyskano rezultat gorszy.

ID	Sensitivity	Specificity	F1	MCC	Balanced Accuracy	AUC
2	0.628	0.758	0.551	0.361	0.693	0.753

Tablica 14: Regresja logistyczna na zbiorze testowym z wykorzystaniem kolumn istotnych

6.3 Maszyna wektorów nośnych (SVM)

W celu przetestowania algorytmu SVM wykorzystano pakiet e1071 [10]. W tabeli 15 przedstawiono oraz opisano zastosowane parametry algorytmu.

Parametr	**Wartość**	**Opis**
Rodzaj jądra	linear, radial	Rodzaj jądra stosowany w modelu.
Koszt	1, 2,5, 5	Parametr kary za przekroczenie marginesu
ROSE	True, False	Flaga wskazująca na użycie algorytmu ROSE w fazie preprocessingu.
Ważenie klas	True, False	Flaga wskazująca na użycie ważenia podczas uczenia.
Imputacja	True	Flaga wskazująca za użycie imputacji średnią lub modalną w fazie preprocessingu

Tablica 15: Opis parametrów modelu SVM

6.3.1 Wyniki dla wybranych konfiguracji parametrów

Podobnie jak w przypadku lasu losowego wybór parametrów został podzielony na grupy, gdzie ważenie i wykorzystanie metody ROSE nie występują jednocześnie. Wyniki z k-krotnej walidacji zostały przedstawione w tabelach 16 oraz 17. Jednakże, w przypadku pierwszej grupy testów napotkaliśmy na problem (albo biblioteki od SVM albo od k-krotnej walidacji albo nasz własny), który objawiał się „odwrotną" predykcją tj. predykcja w postaci prawdopodobieństwa przynależności do klasy była tak naprawdę dopełnieniem do jedynki faktycznej (oczekiwanej) wartości. W związku z tym, że ów wyniki były we wspomnianej postaci dało je się porównać z innymi oraz między sobą (im mniejsza wartość AUC tym lepszy model; wykres ROC również w tym przypadku był odwrócony tj. odbity wzdłuż przekątnej przechodzącej przez początek układu współrzędnych).

ID	**Jądro**	**Koszt**	**ROSE**	**Wagi**	**Imputacja**	**F1**	**MCC**	**Balanced Accuracy**	**AUC**
6	radial	5	False	True	True	0.291	-0.242	0.383	0.325
12	radial	5	False	False	True	0.291	-0.242	0.383	0.329
4	radial	5	False	True	True	0.294	-0.237	0.386	0.333
10	radial	5	False	False	True	0.294	-0.237	0.386	0.334
2	radial	5	False	True	True	0.301	-0.237	0.393	0.344

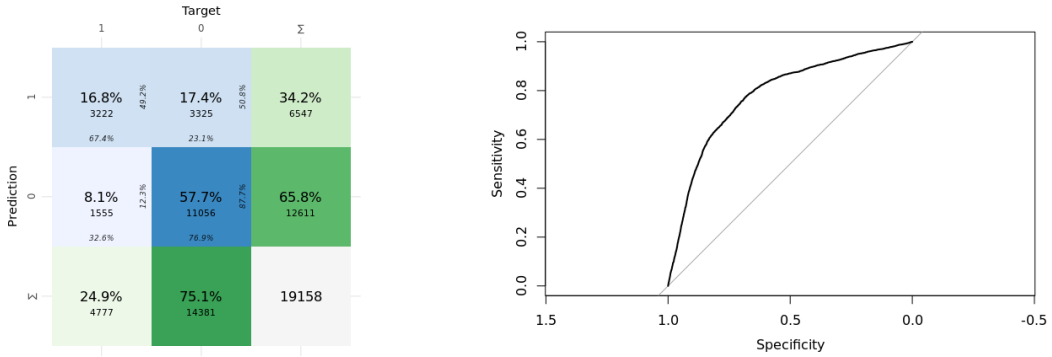
Tablica 16: Wyniki ewaluacji SVM k-krotnej walidacji krzyżowej (zestaw 1)

Jak można zauważyć w wynikach, przewagę mają modele z wyższymi parametrami kosztu, o nieliniowych jądrach oraz wykorzystujących algorytm ROSE lub wagi.

ID	Jądro	Koszt	ROSE	Wagi	Imputacja	F1	MCC	Balanced Accuracy	AUC
6	radial	5	True	False	True	0.569	0.404	0.722	0.782
4	radial	2,5	True	False	True	0.569	0.404	0.722	0.780
5	linear	5	True	False	True	0.569	0.404	0.723	0.778
1	linear	1	True	False	True	0.571	0.406	0.724	0.778
2	radial	1	True	False	True	0.565	0.399	0.719	0.777

Tablica 17: Wyniki ewaluacji SVM k-krotnej walidacji krzyżowej (zestaw 2)

Najlepszy pod względem metryki AUC okazał się model o id 6 z zestawu 2. Krzywa ROC oraz macierz pomyłek znajduje się na rysunku 5b. Model ten również wytrenowano na całym zbiorze trenującym oraz przetestowano na zbiorze testowym wyniki znajdują się w tabeli 18 a wykresy na rysunku 6. Dodatkowo w tabeli 19 przedstawiono wyniki dla najlepszego modelu badanego na danych z ograniczoną liczbą kolumn metodą Boruta. Jak można zauważyć wyniki te są gorsze w porównaniu do najlepszego modelu trenowanego na całym zbiorze danych ze wszystkimi kolumnami.



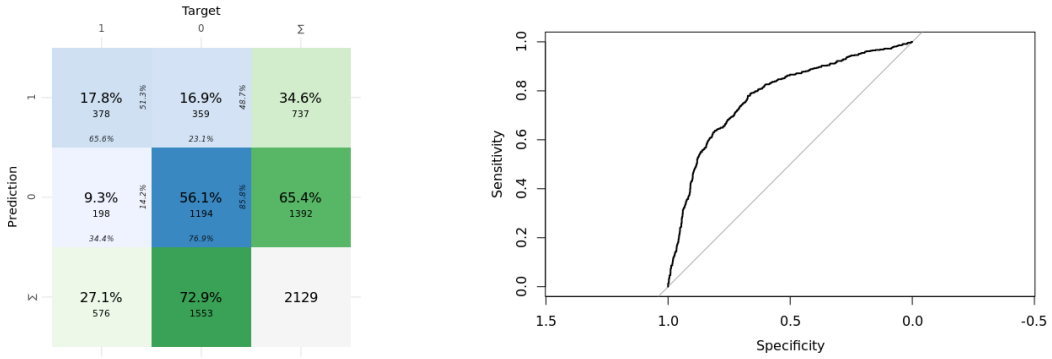
(a) Macierz pomyłek

(b) Krzywa ROC

Rysunek 5: Wyniki ewaluacji SVM dla k-krotnej walidacji krzyżowej

ID	Jądro	Koszt	ROSE	Wagi	Imputacja	F1	MCC	Balanced Accuracy	AUC
6	radial	5	True	False	True	0.576	0.397	0.723	0.779

Tablica 18: Wyniki ewaluacji SVM na zbiorze testowym



(a) Macierz pomyłek

(b) Krzywa ROC

Rysunek 6: Wyniki ewaluacji SVM na zbiorze testowym

ID	**Jądro**	**Koszt**	**ROSE**	**Wagi**	**Imputacja**	**F1**	**MCC**	**Balanced Accuracy**	**AUC**
2	radial	1	True	False	True	0.549	0.360	0.692	0.747

Tablica 19: Wyniki ewaluacji SVM na zbiorze testowym z wykorzystaniem kolumn istotnych

6.4 Gradient boosting

W celu przetestowania algorytmu gradient boosting wykorzystano pakiet xgboost [6]. W tabeli 20 przedstawiono oraz opisano zastosowane parametry algorytmu.

Parametr	**Wartość**	**Opis**
Rundy	1, 10, 100, 1000	Maksymalna liczba drzew.
Głębokość	1, 5, 10, 50	Maksymalna głębokość drzewa.
ROSE	True, False	Flaga wskazująca na użycie algorytmu ROSE w fazie preprocessingu.
Imputacja	True	Flaga wskazująca za użycie imputacji średnią lub modalną w fazie preprocessingu.
Ważenie klas	True, False	Flaga wskazująca na użycie ważenia podczas uczenia.

Tablica 20: Opis parametrów modelu gradient boosting

Jako współczynnik wag przyjęto 3.01, który został obliczony jako iloraz sumy przykładów negatywnych do sumy przykładów pozytywnych. Liczba rund oraz głębokość zostały sprawdzone dla wartości większych oraz mniejszych od wartości domyślnych, które wynosiły odpowiednio 100 oraz 6. Pozostałe parametry pozostały domyślne tzn. współczynnik $\eta = 0.3$, $\gamma = 0$.

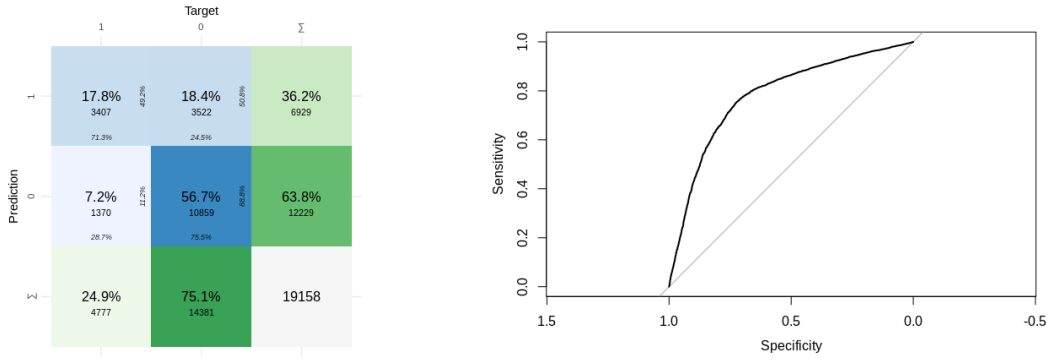
6.4.1 Wyniki dla wybranych konfiguracji parametrów

Spośród wszystkich przetestowanych modeli w k-krotnej walidacji krzyżowej z wykorzystaniem wszystkich atrybutów w tabeli 21 przedstawiono 7 z największą wartością AUC wraz z wynikami metryk podczas walidacji krzyżowej.

ID	**Rundy**	**Głębokość**	**ROSE**	**Imputacja**	**Wagi**	**F1**	**MCC**	**Balanced Accuracy**	**AUC**
4	1000	1	False	True	True	0.582	0.421	0.734	0.783
20	1000	1	False	True	False	0.421	0.319	0.626	0.782
3	100	1	False	True	True	0.572	0.414	0.721	0.780
19	100	1	False	True	False	0.396	0.301	0.615	0.780
6	10	5	False	True	True	0.568	0.406	0.720	0.778
7	100	10	False	True	True	0.582	0.421	0.735	0.777
7	100	5	True	True	False	0.562	0.398	0.713	0.773

Tablica 21: Wyniki k-krotnej walidacji krzyżowej

Analizując tabele 21 można zauważyć, że modele z wagami klas nie tylko osiągnęły rezultat lepszy od metody ROSE ale okazały się również ogólnie najlepsze. Podnosząc takie metryki jak dokładność czy wskaźnik F1 do wartości 0.734 oraz 0.582 odpowiednio. Niespodzianką nie okazały się lepsze wyniki uzyskane dla zbioru zbalansowanego od niezbalansowanego. Należy wspomnieć, że nasz zbiór nie należał do skrajnego przypadku (stosunek 3:1 dla klasy 0), co tylko podkreśla istotę stosowania metod balansujących. Najlepszym modelem okazał się ten z duża liczbą drzew (1000) ale jednocześnie płaskich (głębokość 1). Następnie dla niego (model nr 4) obliczono macierz pomyłek (rys. 7a) oraz wykreślono krzywą ROC (rys. 7b).



(a) Macierz pomyłek (b) Krzywa ROC

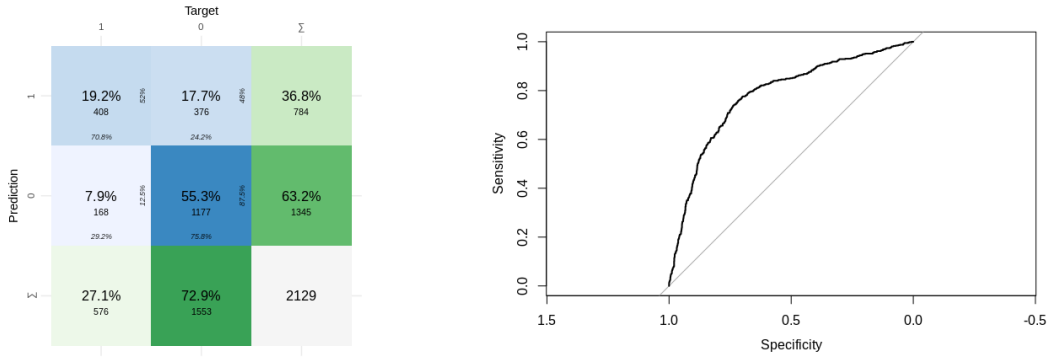
Rysunek 7: Gradient boosting dla k-krotnej walidacji krzyżowej

Następnie został on wytrenowany na pełnym zbiorze treningowym oraz przetestowany na zbiorze ewaluacyjnym. W tabeli 22 przedstawiono otrzymane wyniki.

ID	**Sensitivity**	**Specificity**	**F1**	**MCC**	**Balanced Accuracy**	**AUC**
4	0.708	0.758	0.6	0.429	0.733	0.78

Tablica 22: Wyniki ewaluacji na zbiorze testowym

Macierz pomyłek oraz krzywą ROC przedstawiono na rysunku 8a oraz 8b.



(a) Macierz pomyłek (b) Krzywa ROC

Rysunek 8: Gradient boosting na zbiorze ewaluacyjnym

Rezultaty uzyskane na zbiorze testowym okazały się zbliżone do tych z walidacji krzyżowej (AUC równe 0.78, dokładność na poziomie 0.73).

W celu zbadania poprawy modelu, przeprowadzono kolejne uczenie na pełnym zbiorze tym razem wykorzystując atrybuty skategoryzowane przez algorytm Boruta za istotne. Analizując tabelę 23 można zauważyć pogorszenie w porównaniu do poprzedniego modelu we wszystkich badanych metrykach.

ID	Sensitivity	Specificity	F1	MCC	Balanced Accuracy	AUC
2	0.628	0.752	0.547	0.354	0.690	0.746

Tablica 23: Gradient boosting na zbiorze testowym z wykorzystaniem kolumn istotnych

7 Podsumowanie

W przeprowadzonej analizie uzyskano podobne rezultaty dla wszystkich algorytmów. Dokładność predykcji modeli zawierała się w przedziale $0.76 - 0.79$, osiągając wartość podobną do tych uzyskanych przez uczestników konkursu w serwisie Kaggle (średni wynik wahał się w okolicach 80%). Najgorszym klasyfikatorem okazał się model regresji logistycznej osiągając dokładność na zbiorze testowym na poziomie 0.734 i 0.753 dla odpowiednio wszystkich atrybutów i istotnych. Analizując jego macierz pomyłek (rys. 4a) można zauważyć predykcję klasy 1 na poziomie 0.579, a klasy 0 na 0.85. Najlepsze wyniki uzyskał algorytm gradient boosting osiągając dokładność na poziomie 0.78 oraz charakteryzując się 0.70 poprawnością predykcji klasy 1 oraz 0.758 klasy 0.

Warto wspomnieć o czasie wykonania poszczególnych algorytmów. Modele XGBoost charakteryzowały się najkrótszym czasem uczenia. Natomiast najdłużej trwało uczenie modeli lasu losowego.

W wykonanym projekcie udało się spełnić wszystkie wymagania zadania projektowego. Z wykorzystaniem dobrych praktyk stworzono modele klasyfikacji wykorzystując 4 różne algorytmy. Przy pomocy k-krotnej walidacji krzyżowej sprawdzono wiele kombinacji parametrów oraz wybrano te, które osiągnęły najlepsze wartości metryk. Następnie przy użyciu zbioru treningowego oraz testowego przeprowadzono uczenie na pełnym zbiorze oraz ewaluacje końcową modelu. Analiza wyników umożliwiła porównanie modeli oraz weryfikacje ich skuteczności, dzięki czemu informacje teoretyczne zostały zaobserwowane w praktyce.

Literatura

[1] Boruta. *Boruta: Feature selection with the Boruta algorithm*, 2021. URL https://www.rdocumentation.org/packages/Boruta/versions/5.2.0/topics/Boruta.

[2] dummyVars. *Create A Full Set of Dummy Variables*, 2021. URL https://www.rdocumentation.org/packages/caret/versions/6.0-86/topics/dummyVars.

[3] glm. *fit a GLM with lasso or elasticnet regularization*, 2021. URL https://www.rdocumentation.org/packages/glmnet/versions/4.1-1/topics/glmnet.

[4] Kaggle. Hr analytics: Job change of data scientists, 2020. URL https://www.kaggle.com/arashnic/hr-analytics-job-change-of-data-scientists.

[5] Pakiet cvms. Cross-validation for model selection, 2021. URL https://cran.r-project.org/web/packages/cvms/cvms.pdf.

[6] Pakiet xgboost. Extreme gradient boosting, 2021. URL https://cran.r-project.org/web/packages/xgboost/xgboost.pdf.

[7] Parametr K. *Applied Predictive Modeling*, 2021. URL https://machinelearningmastery.com/k-fold-cross-validation/.

[8] Random Forest. *Classification and Regression with Random Forest*, 2021. URL https://www.rdocumentation.org/packages/randomForest/versions/4.6-14/topics/randomForest.

[9] Sample. *Over-sampling, under-sampling, combination of over- and under-sampling*, 2021. URL https://www.rdocumentation.org/packages/ROSE/versions/0.0-3/topics/ovun.sample.

[10] SVM. Support vector machines, 2021. URL https://cran.r-project.org/web/packages/e1071/vignettes/svmdoc.pdf.