Heart failure

Grzegorz Molak, Maciek Wasiluk

2023-08-28

Dane

Heart failure clinical records. (2020). UCI Machine Learning Repository. https://doi.org/10.24432/C5Z89R.

Aplikacja Shiny https://grzegorzmolak.shinyapps.io/HeartFailure/

Dane przedstawiają dane medyczne 299 pacjentów ze zdiagnozowaną niewydolnością serca.

Na podstawie ekspolarcji danych oraz zbudowania modeli przeprowadzona zostanie analiza możliwości predykcji niewydolności serca oraz wyodrębnienie głównych cech za nią odpowiedzialnych.

Eksploracja danych

```
summary <- data.frame()
for(col in colnames(df)){
  stats <- list(feature = col,
                n_unique = n_distinct(df[[col]]),
                NaNs = sum(is.na(df[[col]])),
                min=min(df[[col]]),
                max=max(df[[col]]))
  summary <- rbind(summary, stats)
}
print(summary)
```

Liczba unikalnych i brakujących wartości, ich zakres

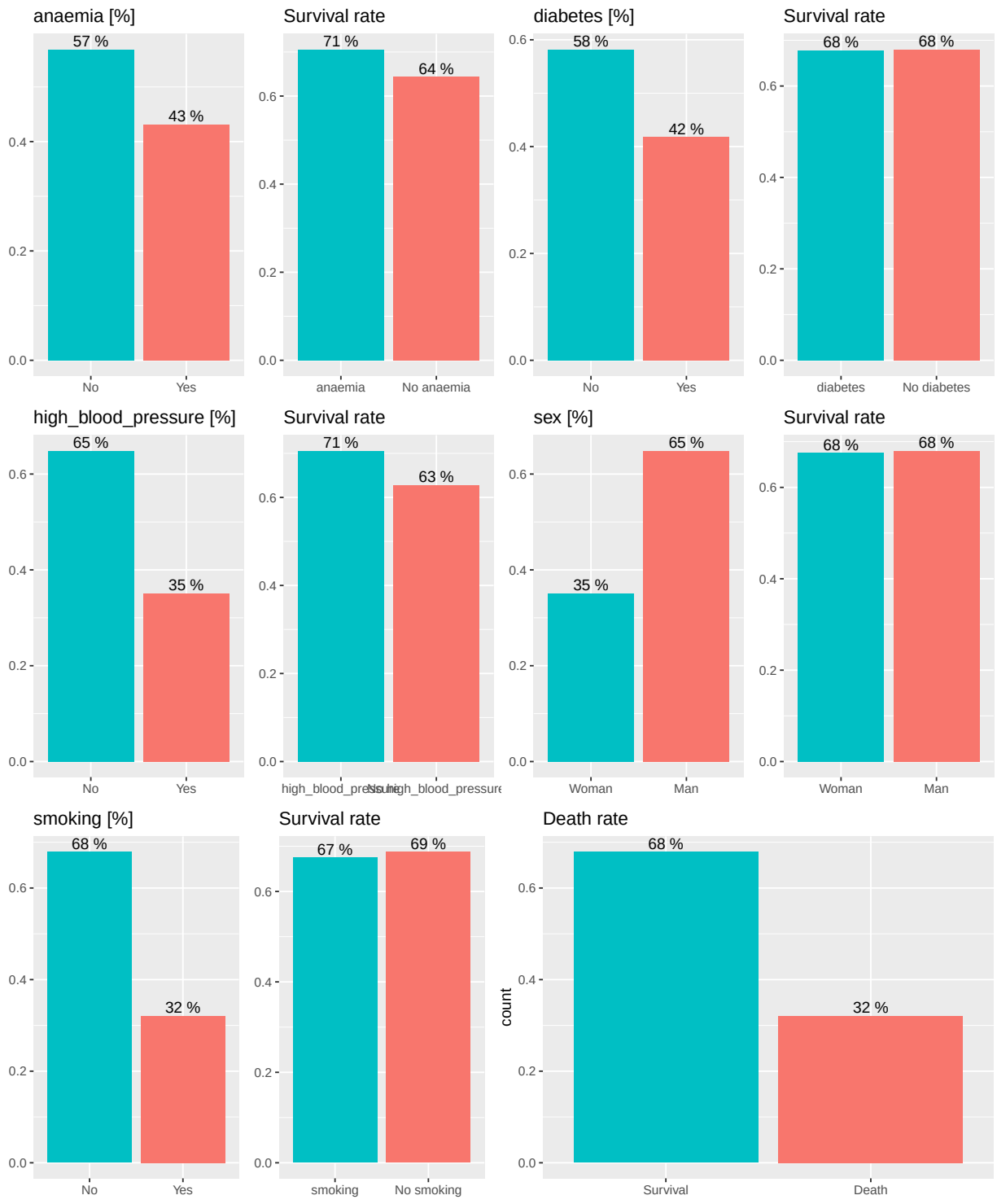
```
##                    feature n_unique NaNs     min      max
## 1                      age       47    0    40.0     95.0
## 2                  anaemia        2    0     0.0      1.0
## 3  creatinine_phosphokinase      208    0    23.0   7861.0
## 4                 diabetes        2    0     0.0      1.0
## 5         ejection_fraction       17    0    14.0     80.0
## 6       high_blood_pressure        2    0     0.0      1.0
## 7                 platelets      176    0 25100.0 850000.0
## 8          serum_creatinine       40    0     0.5      9.4
## 9              serum_sodium       27    0   113.0    148.0
## 10                      sex        2    0     0.0      1.0
## 11                  smoking        2    0     0.0      1.0
```

```
## 12                    time      148    0     4.0    285.0
## 13            DEATH_EVENT        2    0     0.0      1.0
```

`time` jako czas obserwacji pacjenta po zaobserwowaniu niewydolności serca jest naszym zdaniem niewygodna z powodu braku dokładniej wiedzy o znaczeniu tej cechy

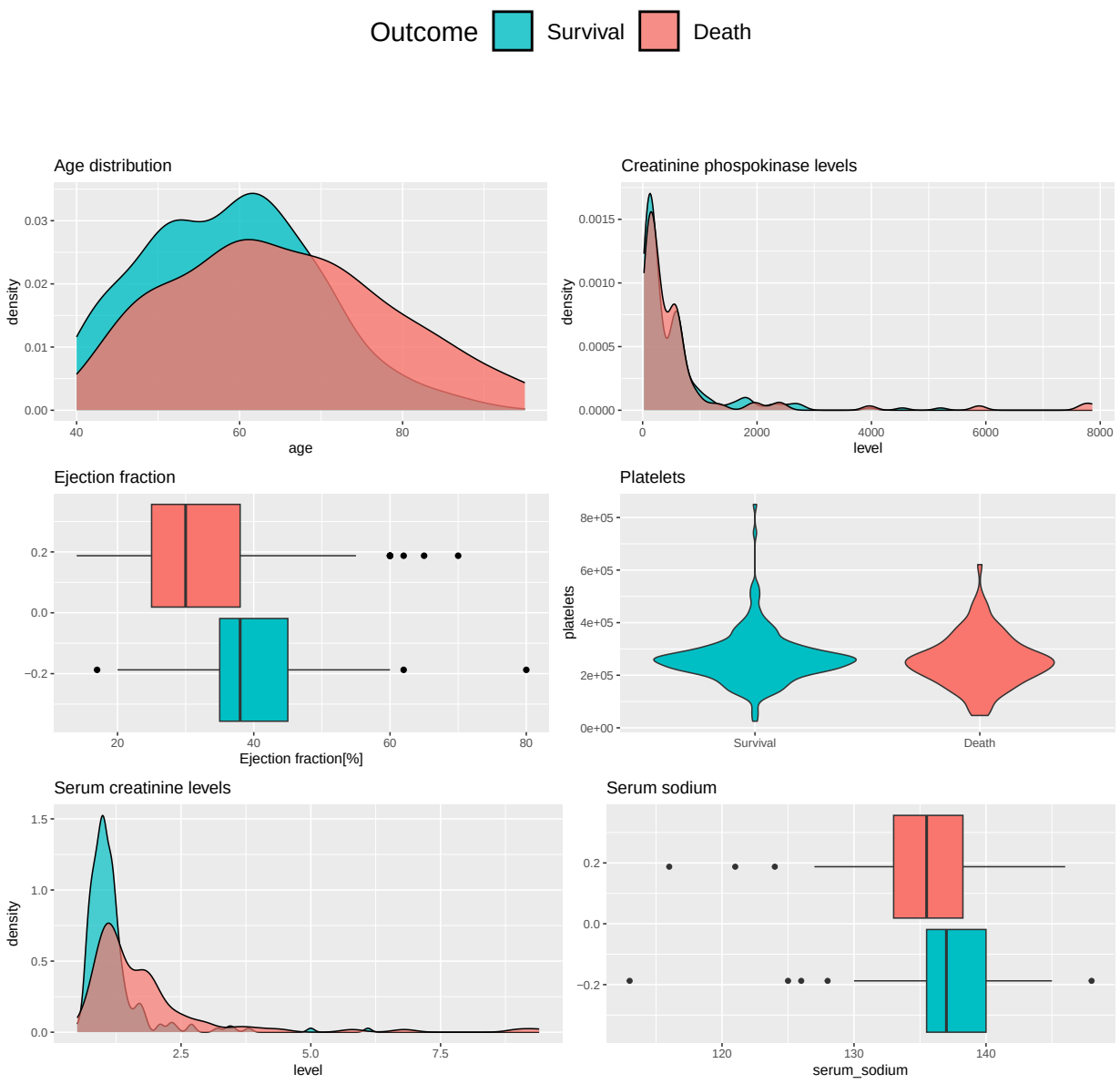
```
df <- df %>% select(-"time")
```

Cechy o rozkładzie binarnym



Różnice w przeżywalności dla różnych wartości cech binarnych nie są zbyt duże, największe różnice widoczne są dla anemii oraz nadciśnienia, lecz nawet w tym przypadku jest to około 8 punktów procentowych

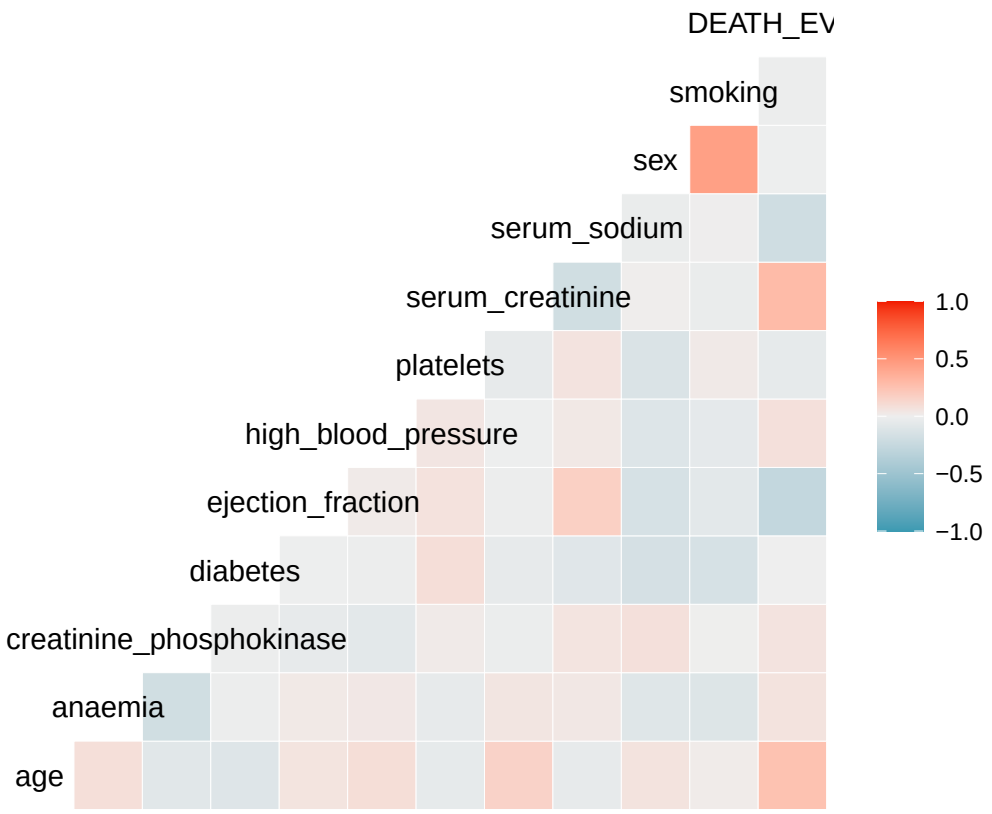
Pozostałe cechy



Można zauważyć, że w przypadku tych cech, dla każdej z nich można zauważyć pewne różnice w dystrybucji dla pacjentów którzy przeżyli oraz pacjentów zmarłych. Można przypuszczać, że cechy o ciągłych wartościach mogą dostarczyć większych wartości predykcyjnych.

Macierz korelacji Wyznaczymy również 3 cechy o największym module współczynnika korelacji, jako podejrzane o wysoką wartość predykcyjną. Czy współczynnik korelacji jest dodatni czy ujemny - można wywnioskować na podstawie koloru

```
ggcorr(df, method = c("pairwise", "pearson"))
```




```
colnames(cor(df))[order(abs(cor(df)["DEATH_EVENT",1:ncol(df)-1]), decreasing=TRUE)[1:3]]
```

```
## [1] "serum_creatinine"  "ejection_fraction" "age"
```

Modele Zdecydowaliśmy się stworzyć modele regresji logistycznej oraz lasu losowego. Oba radzą sobie całkiem nieźle nawet z niewielką ilością danych: 299, nie wymagają skalowania oraz normalizacji danych, w dodatku las losowy umożliwia zbadanie ważności cech na podstawie średniej zmiany współczynnika Giniego.

Przygotowanie danych Jedyne operacje na danych, aby przygotować je modelu to podział na zbiór treningowy i testowy. Dla wybranych modeli skalowanie i normalizacja danych nie są potrzebne.

```
set.seed(0)
df_classification <- df
df_classification$DEATH_EVENT = factor(df$DEATH_EVENT, levels = c(0, 1))
ind <- createDataPartition(df_classification$DEATH_EVENT, p = 0.8, list = FALSE)
trainingset <- df_classification[ind,]
testingset <- df_classification[-ind,]
```

Wielkość zbioru uczącego: 240 oraz testującego: 59

Udział zmarłych pacjentów wewnątrz zbiorów: (32%, 32%)

```
fit_glm <- glm(DEATH_EVENT ~ ., data = trainingset, family = "binomial")
results_glm <- predict(fit_glm, testingset, type = "response")
```

Regresja logistyczna

```
fit_rf <- randomForest(DEATH_EVENT ~ ., data = trainingset, ntree = 100)
results_rf <- predict(fit_rf, testingset, type = "prob")
```

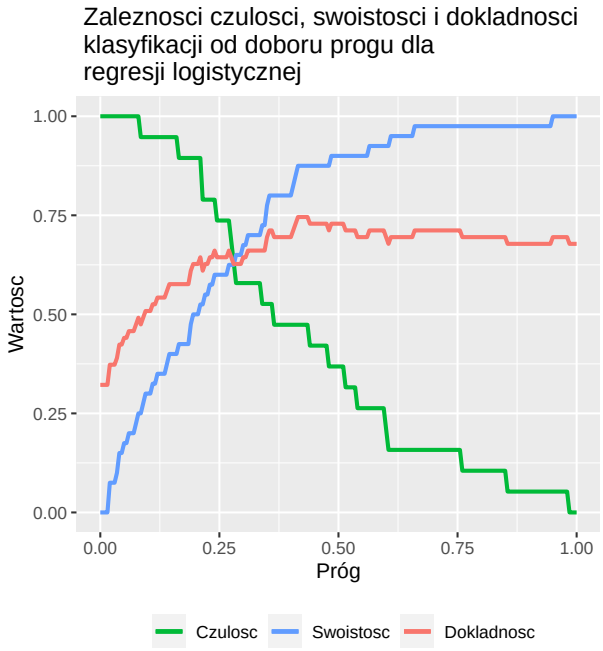
Las losowy

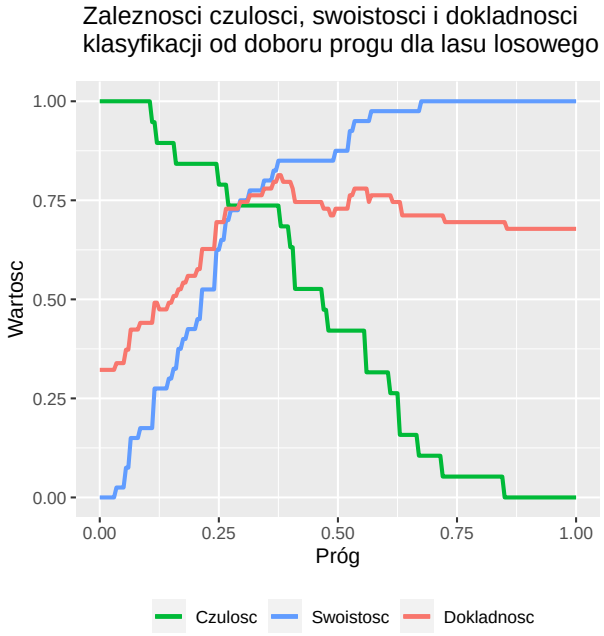
```
accuracy <- function(actual, predicted, threshold)
  {
    mean(as.integer(predicted >= threshold)== actual)
  }

evaluate <- function(prediction_glm, prediction_rf,
                       testing_set)
{
  thresholds <- seq(0,1, by=.005)
  metrics <- data.frame()
  actual <- as.numeric(testing_set$DEATH_EVENT) - 1



  for(threshold in thresholds)
  {
    iter <- c(
              threshold,
              ModelMetrics::auc(actual, prediction_glm),
              ModelMetrics::sensitivity(actual, prediction_glm, threshold),
              ModelMetrics::specificity(actual, prediction_glm, threshold),
              accuracy(actual, prediction_glm, threshold),
              ModelMetrics::precision(actual, prediction_glm, threshold),
              ModelMetrics::recall(actual, prediction_glm, threshold),
              ModelMetrics::f1Score(actual, prediction_glm, threshold),
              ######################################################
              ModelMetrics::auc(actual,prediction_rf[,2]),
              ModelMetrics::sensitivity(actual, prediction_rf[,2], threshold),
              ModelMetrics::specificity(actual, prediction_rf[,2], threshold),
              accuracy(actual, prediction_rf[,2], threshold),
              ModelMetrics::precision(actual, prediction_rf[,2], threshold),
              ModelMetrics::recall(actual, prediction_rf[,2], threshold),
              ModelMetrics::f1Score(actual, prediction_rf[,2], threshold))
    metrics <- rbind(metrics, iter)
```

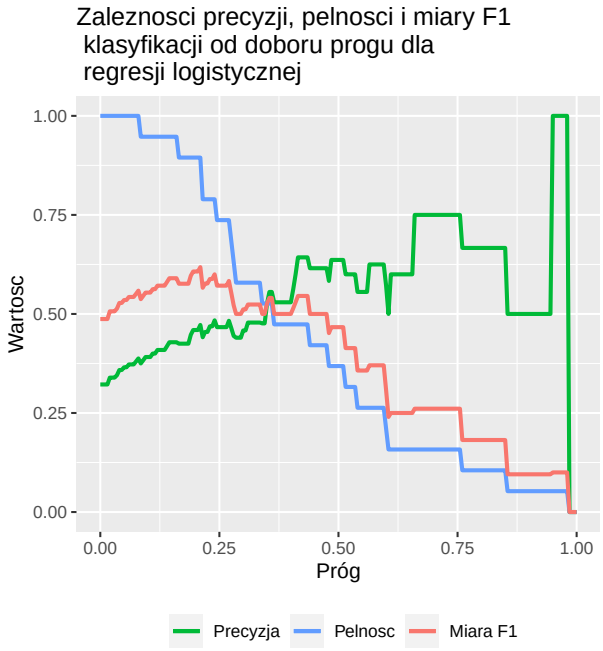
```
  }

  colnames(metrics) <- c("Threshold", "glm_AUC",
                         "glm_Sensitivity", "glm_Specificity",
                         "glm_Accuracy", "glm_Precision",
                         "glm_Recall", "glm_F1_Score",
                         "rf_AUC",
                         "rf_Sensitivity", "rf_Specificity",
                         "rf_Accuracy", "rf_Precision",
                         "rf_Recall", "rf_F1_Score")
return(metrics)
}
```

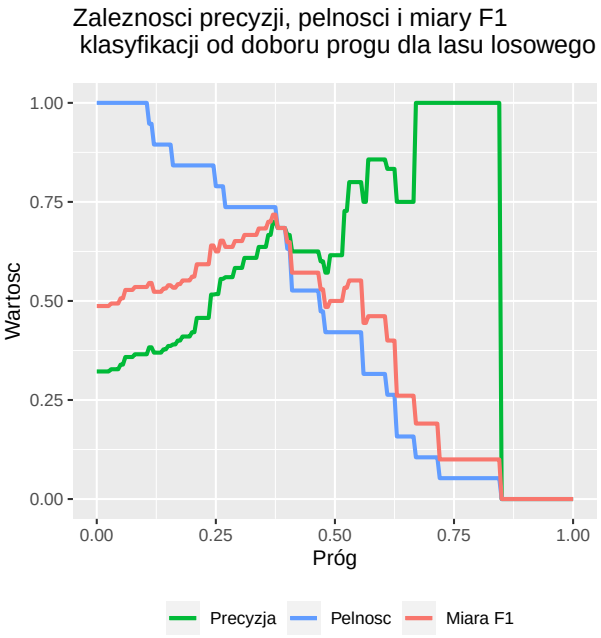
Ewaluacje modeli Wyświetlmy wykresy dla modeli

```
metrics <- evaluate(results_glm, results_rf, testingset)
plot_metrics(metrics)
```

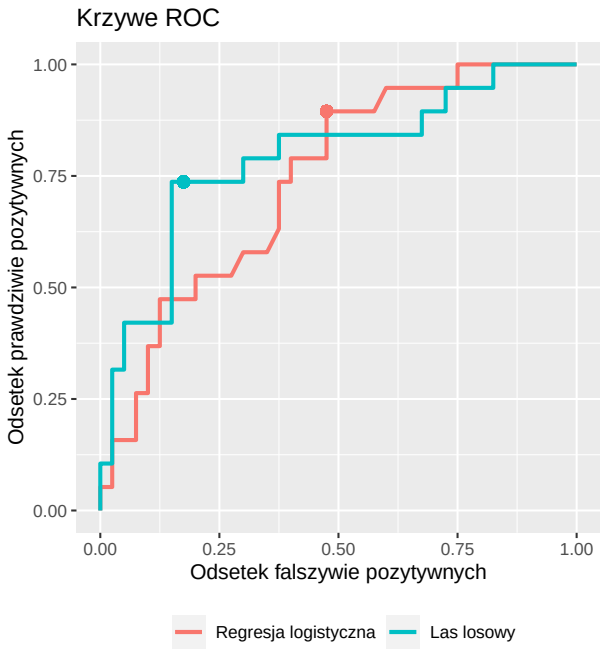


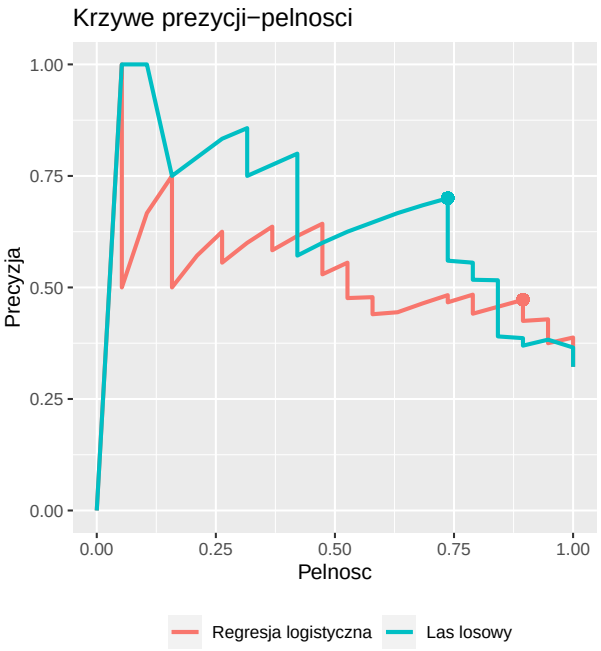







```
plot_roc_prec_recall(metrics)
```





Optymalne wartości progów uzyskamy poprzez znalezienie dla każdego z modeli takiej wartości progów, aby wartość miary F1 była jak największa. Z racji, że miara F1 średnią harmoniczną precyzji oraz pełności, jej najwyższe wartości wskazują w pojedynczej wartości na najlepszy kompromis między precyzją oraz pełnością. Możliwe są również inne metody wyboru optymalnego progu jak na przykład maksymalizacja sumy czułości i swoistości, bądź nawet faworyzowanie jednej miary kosztem innej.

Wartości miar dla wyznaczonych progów zostały oznaczone na krzywych w postaci kół.

```
optimum_glm <- metrics[which.max(metrics$glm_F1_Score),1:8]
optimum_rf <- metrics[which.max(metrics$rf_F1_Score),c(1, 9:15)]
optimum_metrics <- t(data.frame(t(optimum_glm), t(optimum_rf)))
colnames(optimum_metrics) = c("Próg", "AUC", "Czułość", "Swoistość", "Dokładność", "Precyzja", "Pełność
rownames(optimum_metrics) = c("Regresja log.", "Las losowy")
print(t(optimum_metrics))
```

```
##            Regresja log. Las losowy
## Próg           0.2100000  0.3700000
## AUC            0.7421053  0.7953947
## Czułość        0.8947368  0.7368421
## Swoistość      0.5250000  0.8250000
## Dokładność     0.6440678  0.7966102
## Precyzja       0.4722222  0.7000000
## Pełność        0.8947368  0.7368421
## Miara F1       0.6181818  0.7179487
```

Na podstawie wybranej miary ustalone zostały progi decyzyjne na poziomie 0.21 dla regresji logistcznej i 0.37 dla lasu losowego.

Oznacza to, że przewidujemy śmierć pacjenta dla modelu regresji logistycznej, jeżeli na wyjściu modelu otrzymujemy wartość przekraczającą 0.21, podczas gdy dla modelu lasu losowego śmierć pacjenta przewidujemy dopiero powyżej wartości 0.37.

Pozostałe wartości mówią między innymi o tym, że przy wybranych progach, dla kolejno modeli regresji logistycznej i lasu losowego:

- Rozpoznawane jest 89.5% i 73.7% przypadków śmierci pacjenta
- Rozpoznawane jest 52.5% i 82.5% przypadków przeżycia pacjenta
- Prawidłowo zgadywane jest 64.5% i 79,7% przypadków
- Wśród przewidzianych śmierci pacjenta, prawidłowych jest 47,2% i 70% prawidłowych diagnoz

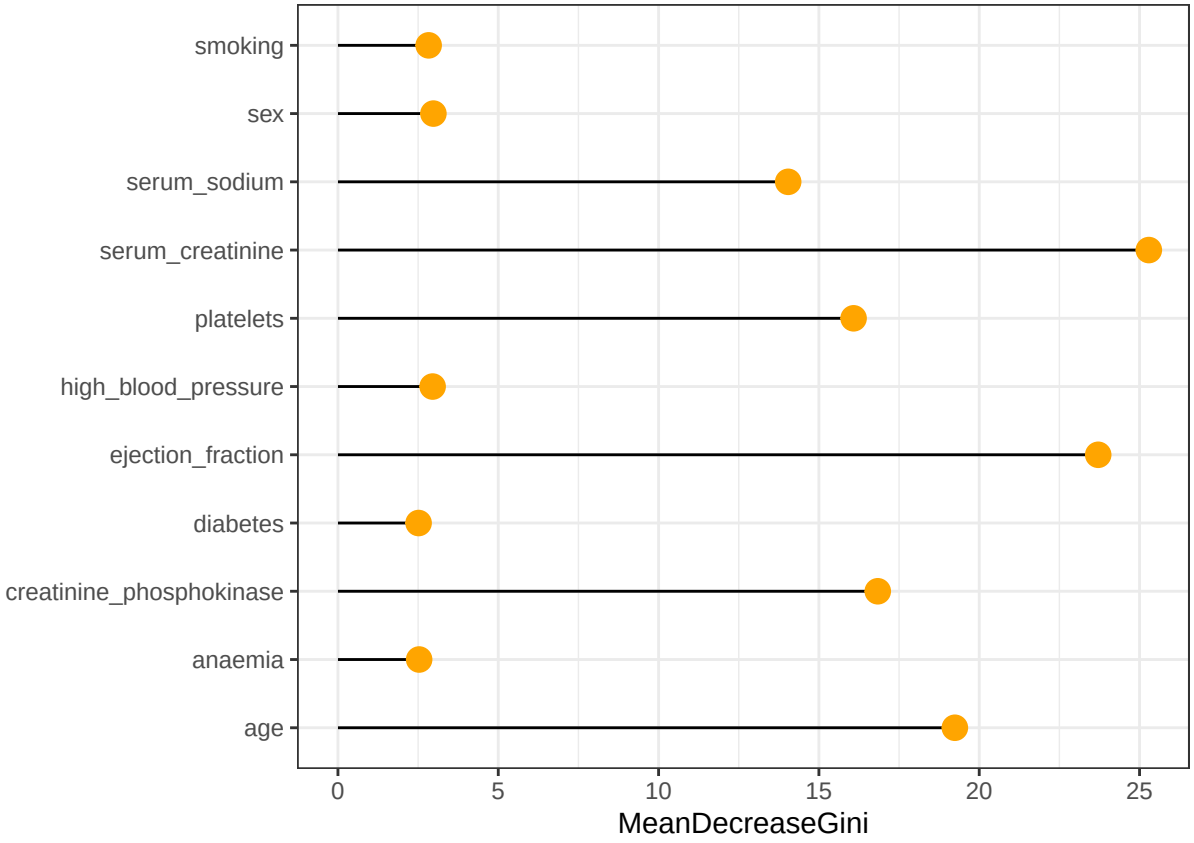
Widoczne jest, że dla wybranych progów, regresja logistyczna jest znacznie bardziej czuła, to znaczy trafniej przewiduje śmierć pacjenta, lecz powoduje to, że bardzo rzadko przewiduje przeżycie. Widoczne jest to przez niską wartość precyzji modelu, która wskazuje na dużą ilość "fałszywych pozytywów" Z kolei las losowy zdaje się znajdywać pewien kompromis między czułością i swoistością. W zależności od potrzebnej miary progi decyzyjne można ustawić inaczej, na przykład w przypadku gdy zależy nam, aby test wykrywał przypadki pozytywne z pewną nie mniejszą od pewnej wartości pewnością, należałoby znaleźć wartość progu, dla której przykładowo wartość czułości wynosi 0.95%

Ranking cech - redukcja wymiarowości

Las losowy Z modelu lasu losowego nauczonego na całym zbiorze danych zbierzemy informacje o ważności poszczególnych cech przy kwalifikacji

```
imp_rf <- randomForest(DEATH_EVENT ~ ., data = df_classification, ntree = 100, importance=TRUE)
importances = data.frame(rownames(imp_rf$importance), imp_rf$importance[,4])
colnames(importances) <- c("Feature", "MeanDecreaseGini")

importances %>%
```

```
      ggplot( aes(x=Feature, y=MeanDecreaseGini)) +
      geom_segment( aes(xend=Feature, yend=0)) +
      geom_point( size=4, color="orange") +
      coord_flip() +
      theme_bw() +
      xlab("")
```



RFE - recursive feature elimination Metoda RFE wyznaczania cech(`feature selection`) polega na wykorzystaniu innego algorytmu uczenia maszynowego - w naszym przypadku modelu lasu losowego oraz pewnej funkcji celu - w naszym przypadku dokładności, do wybrania najlepszego podzbioru cech o wybranym rozmiarze na podstawie podanych wcześniej metryk.

Algorytm rozpoczyna działanie tworząc model(lasu losowego) korzystając ze wszystkich dostępnych cech, wybiera na jego podstawie najmniej potrzebną cechę, po czym uruchamiany jest znowu, z nowym, zmniejszonym zestawem cech - algorytm działa na zasadzie rekurencji.

```
set.seed(1)
control_rfe = rfeControl(functions = rfFuncs)
train_x <- df_classification %>% select(-"DEATH_EVENT")
train_y <- df_classification$DEATH_EVENT
result_rfe = rfe(x = train_x,
                 y = train_y,
                 sizes = c(2:3),
                 rfeControl = control_rfe,
```

```
                metric = "Accuracy")
result_rfe$optVariables
```

```
## [1] "serum_creatinine"  "ejection_fraction" "age"
```

Redukcja wymiarowości

Z przeprowadzonych analiz wynika, że cechami niosącymi największą wartość dla modeli są `serum_creatinie`, `ejection_fraction` oraz `age` Na podstawie tych założeń zbudowane zostaną te same modele co poprzednio, jednak przy wykorzystaniu danych jedynie z wybranych kolumn

```
subset <- df_classification %>% select("serum_creatinine", "ejection_fraction", "age", "DEATH_EVENT")
trainingsubset <- subset[ind,]
testingsubset <- subset[-ind,]
```

Wybór podzbiorów

```
fit_3glm <- glm(DEATH_EVENT ~ ., data = trainingsubset, family = "binomial")
results_3glm <- predict(fit_3glm, testingsubset, type = "response")
```

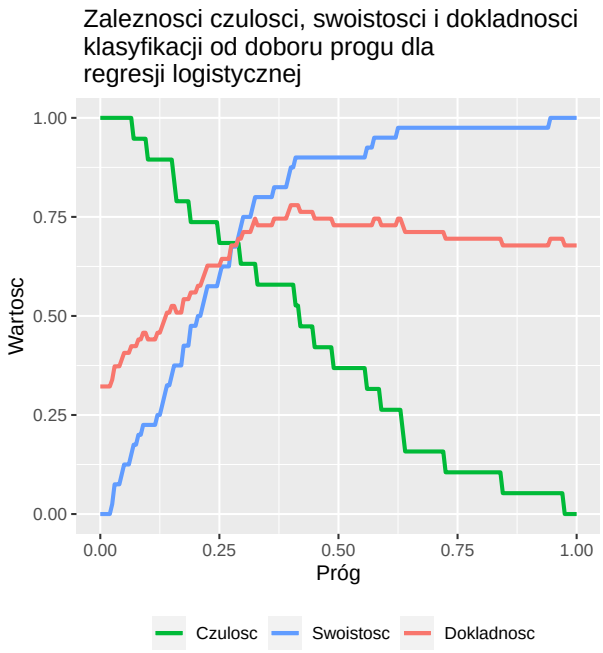
Regresja logistyczna

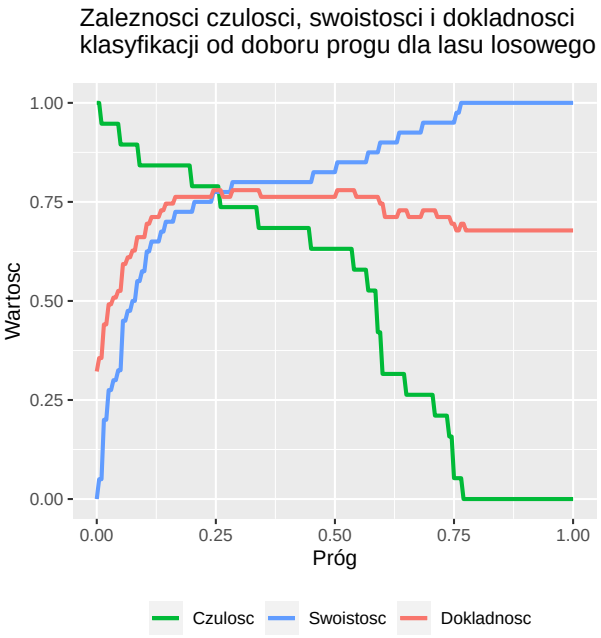
```
fit_3rf <- randomForest(DEATH_EVENT ~ ., data = trainingsubset, ntree = 100)
results_3rf <- predict(fit_3rf, testingsubset, type = "prob")
```

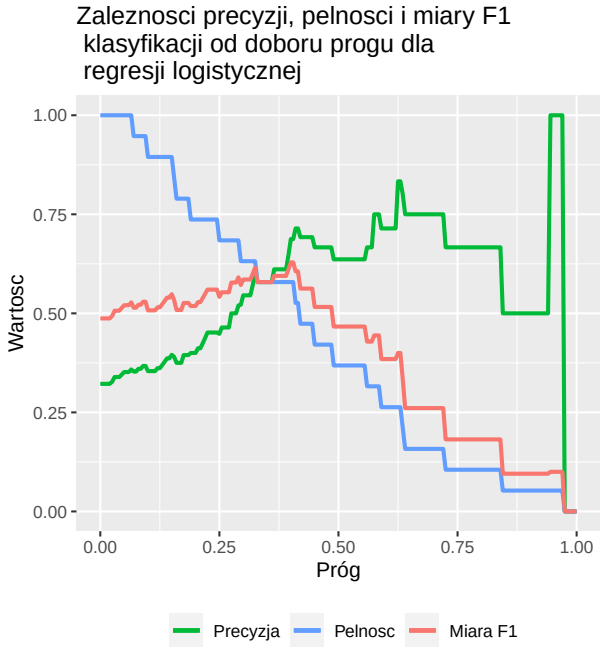
Las losowy

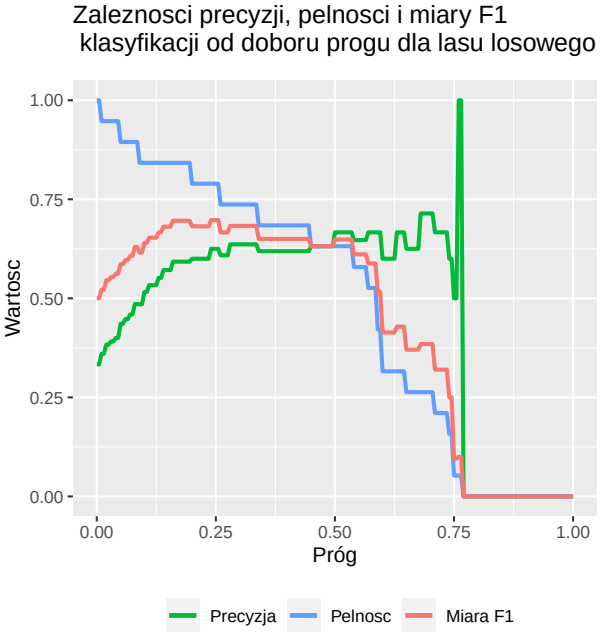
```
metrics_subset <- evaluate(results_3glm, results_3rf, testingsubset)
plot_metrics(metrics_subset)
```

Metryki

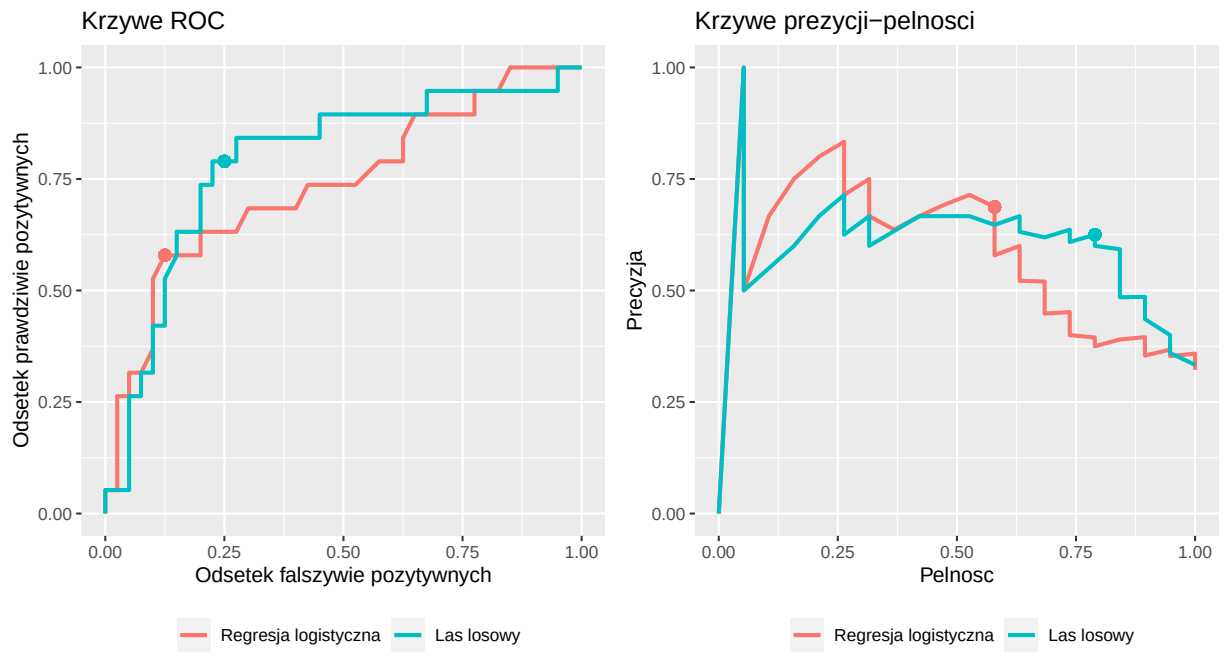









```
plot_roc_prec_recall(metrics_subset)
```




```
optimum_3glm <- metrics_subset[which.max(metrics_subset$glm_F1_Score),1:8]
optimum_3rf <- metrics_subset[which.max(metrics_subset$rf_F1_Score),c(1, 9:15)]
optimum_metrics <- rbind(optimum_metrics, t(data.frame(t(optimum_3glm), t(optimum_3rf))))
rownames(optimum_metrics)[c(3,4)] = c("Regresja zred.", "Las losowy zred.")
print(t(optimum_metrics))
```

Porównanie metryk dla całego zbioru i 3 wybranych cech(zredukowany zbiór cech)

```
##            Regresja log. Las losowy Regresja zred. Las losowy zred.
## Próg           0.2100000  0.3700000      0.4000000        0.2400000
## AUC            0.7421053  0.7953947      0.7394737        0.8019737
## Czułość        0.8947368  0.7368421      0.5789474        0.7894737
## Swoistość      0.5250000  0.8250000      0.8750000        0.7500000
## Dokładność     0.6440678  0.7966102      0.7796610        0.7627119
## Precyzja       0.4722222  0.7000000      0.6875000        0.6250000
## Pełność        0.8947368  0.7368421      0.5789474        0.7894737
## Miara F1       0.6181818  0.7179487      0.6285714        0.6976744
```

Wyniki nowego modelu regresji logistycznej zdają się być przeciwieństwem modelu opartego na pełnym zestawie cech. Uzyskał on znacznie większą swoistość kosztem znacznie mniejszej czułości.

Nowy model lasu losowego nie różni się tak diametralnie od poprzedniego w porównaniu do regresji logistycznej. Jest on nieco bardziej czuły, kosztem niewielkiej utraty swoistości oraz precyzji.

Z przeprowadzonych analiz wynika, że przewidywanie przeżywalności na podstawie 3 najważniejszych cech może być co najmniej równie dobra, jak na podstawie wszystkich. Może to wynikać z tego, że zbiór posiada bardzo małą ilość danych(299 rekordów), w związku z czym modelom ciężko jest dopasować się do danych, które mają zbyt dużą liczbę cech.