Regresja logistyczna

Zacznijmy od danych dotyczących tego czy studenci zostali przyjęci na studia.

```
admissions <- read.csv("http://www.ats.ucla.edu/stat/data/binary.csv")
knitr::kable(head(admissions))
```

admit	gre	gpa	rank
0	380	3.61	3
1	660	3.67	3
1	800	4.00	1
1	640	3.19	4
0	520	2.93	4
1	760	3.00	2

Interesuje nas klasyfikacja obserwacji do jednej z dwóch grup. Np. dostał się/nie dostał się, chory/zdrowy. Jest to co prawda zupełnie inne zadanie niż regresja, ale jesteśmy w stanie dokonać klasyfikacji w oparciu o model liniowy. Takie bezpośrednie podejście nazywa się **Linear Discriminant Analysis**. Nie będzie ono jednak przedmiotem naszego zainteresowania.

W dalszej części zajęć będziemy zajmować się danymi dotyczącymi badania chorób serca. Pochodzą one z jednego z najbardziej znanych ekperymentów epidmiologicznych, prospektywnego badania w miejscowości Framingham (MA). Od lat 20-tych, przez kilkadziesiąt lat, raz na dwa lata mieszkańcy miasteczeka byli przepytywani i badany był ich stan zdrowia.

```
framingham=read.csv("framingham.csv")
summary(framingham)
```

```
##       male             age          education     currentSmoker
##  Min.   :0.0000   Min.   :32.00   Min.   :1.000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:42.00   1st Qu.:1.000   1st Qu.:0.0000
##  Median :0.0000   Median :49.00   Median :2.000   Median :0.0000
##  Mean   :0.4292   Mean   :49.58   Mean   :1.979   Mean   :0.4941
##  3rd Qu.:1.0000   3rd Qu.:56.00   3rd Qu.:3.000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :70.00   Max.   :4.000   Max.   :1.0000
##                                   NA's   :105
##    cigsPerDay         BPMeds        prevalentStroke     prevalentHyp
##  Min.   : 0.000   Min.   :0.00000   Min.   :0.000000   Min.   :0.0000
##  1st Qu.: 0.000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.:0.0000
##  Median : 0.000   Median :0.00000   Median :0.000000   Median :0.0000
##  Mean   : 9.006   Mean   :0.02962   Mean   :0.005896   Mean   :0.3106
##  3rd Qu.:20.000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:1.0000
##  Max.   :70.000   Max.   :1.00000   Max.   :1.000000   Max.   :1.0000
##  NA's   :29       NA's   :53
##     diabetes          totChol          sysBP           diaBP
##  Min.   :0.00000   Min.   :107.0   Min.   : 83.5   Min.   : 48.0
##  1st Qu.:0.00000   1st Qu.:206.0   1st Qu.:117.0   1st Qu.: 75.0
##  Median :0.00000   Median :234.0   Median :128.0   Median : 82.0
##  Mean   :0.02571   Mean   :236.7   Mean   :132.4   Mean   : 82.9
##  3rd Qu.:0.00000   3rd Qu.:263.0   3rd Qu.:144.0   3rd Qu.: 90.0
##  Max.   :1.00000   Max.   :696.0   Max.   :295.0   Max.   :142.5
##                    NA's   :50
##       BMI          heartRate         glucose         TenYearCHD
```

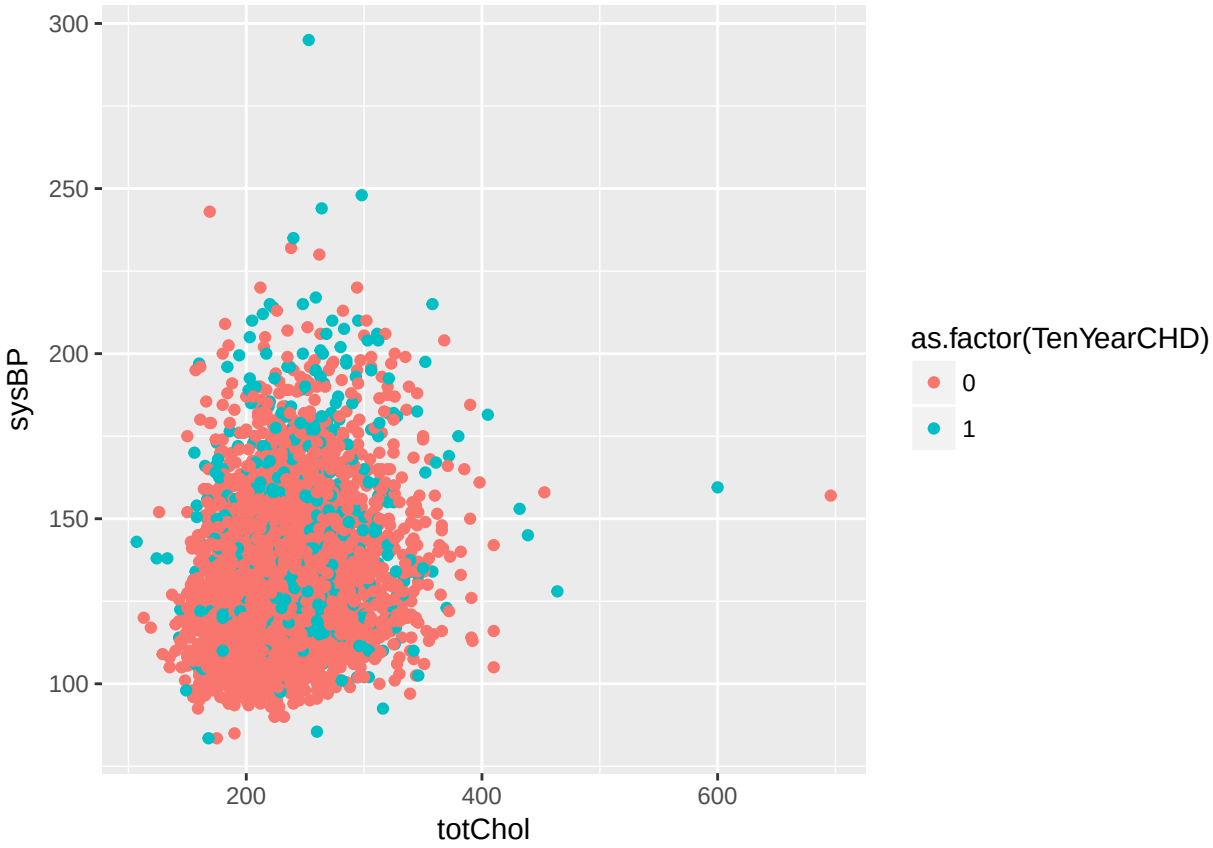
```
##  Min.   :15.54   Min.   : 44.00   Min.   : 40.00   Min.   :0.0000
##  1st Qu.:23.07   1st Qu.: 68.00   1st Qu.: 71.00   1st Qu.:0.0000
##  Median :25.40   Median : 75.00   Median : 78.00   Median :0.0000
##  Mean   :25.80   Mean   : 75.88   Mean   : 81.96   Mean   :0.1519
##  3rd Qu.:28.04   3rd Qu.: 83.00   3rd Qu.: 87.00   3rd Qu.:0.0000
##  Max.   :56.80   Max.   :143.00   Max.   :394.00   Max.   :1.0000
##  NA's   :19      NA's   :1        NA's   :388
```

Naszą zmienną objaśnianą jest ryzyko wystąpienia choroby niedokrwiennej serca (ang. CHD) w perspektywnie 10 lat (*TenYearCHD*).

Zobaczmy jak wyglądają nasze dane:

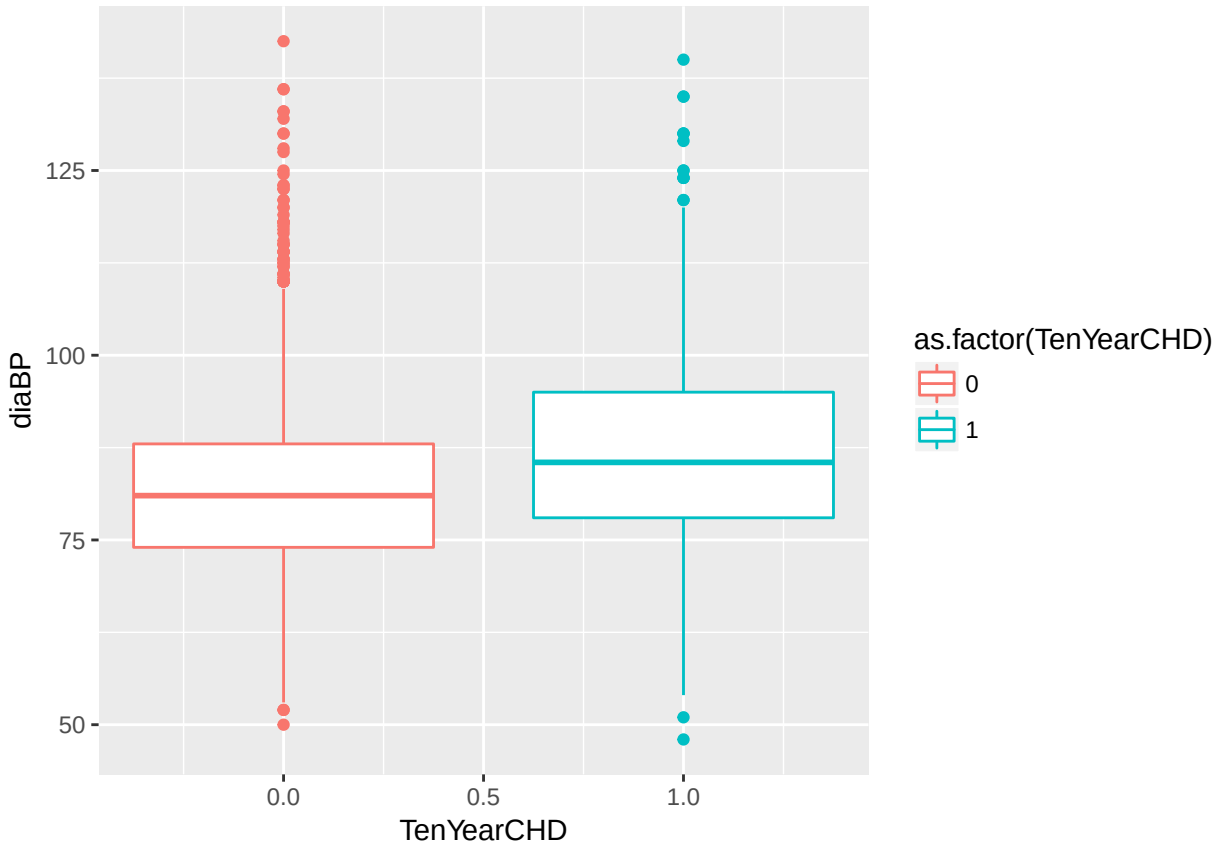
```
ggplot(framingham) +
  geom_point(aes(x=totChol, y=sysBP, color=as.factor(TenYearCHD)))
```

```
## Warning: Removed 50 rows containing missing values (geom_point).
```



Dane są mocno poprzemieszanie, ale jest jakaś nadzieja. Oczywiście podobne wykresy można zrobić dla innych zmiennych.

```
ggplot(framingham) +
  geom_boxplot(aes(x=TenYearCHD, y=diaBP, color=as.factor(TenYearCHD)))
```



W modelu regresji logistycznej określamy p-stwo tego, że zmienna objaśniana przyjmie konkretną wartość:

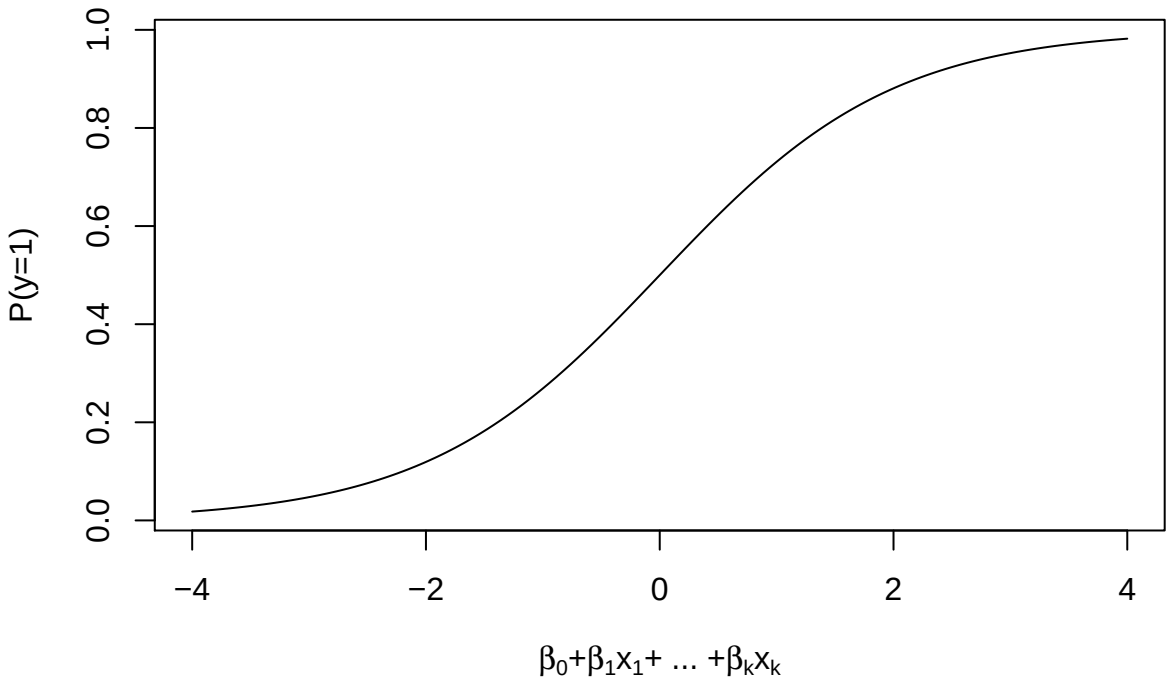
$$P(\textbf{TenYearCHD} = 1) = 1 - P(\textbf{TenYearCHD} = 0) = ???$$

Teraz zaczyna się „modelowanie". Jak zmienne objaśniające (oznaczmy je przez $x_1, x_2, \ldots, x_k$) mają wpływać na zmienną wynikową y?

Chcielibyśmy mieć coś prostego co prowadzi nas do naszego starego znajomego: przekształcenia liniowego: $\beta_0 + \beta_1 x_1 + \ldots \beta_k x_k$. Niestety zbiór wartości tego przekształcenia to cała prosta rzeczywista. Stosujemy więc przekształcenie:

$$P(y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \ldots \beta_k x_k)}}$$

```
x=seq(-4,4, 0.005)
plot(x, 1/(1+exp(-x)), type="l", ylab = "P(y=1)", xlab=expression(paste(beta[0], "+", beta[1], x[1], "+
```



Widziamy, że jeśli funkcja liniowa od zmiennych objaśnianych ma wartość większą niż 0, to bardziej prawdopodobny jest $y = 1$. Wyraźmy wybór między 0 a 1 nie przez p-stwa, ale przez iloraz szans.

$$Odds = \frac{P(y=1)}{P(y=0)}$$

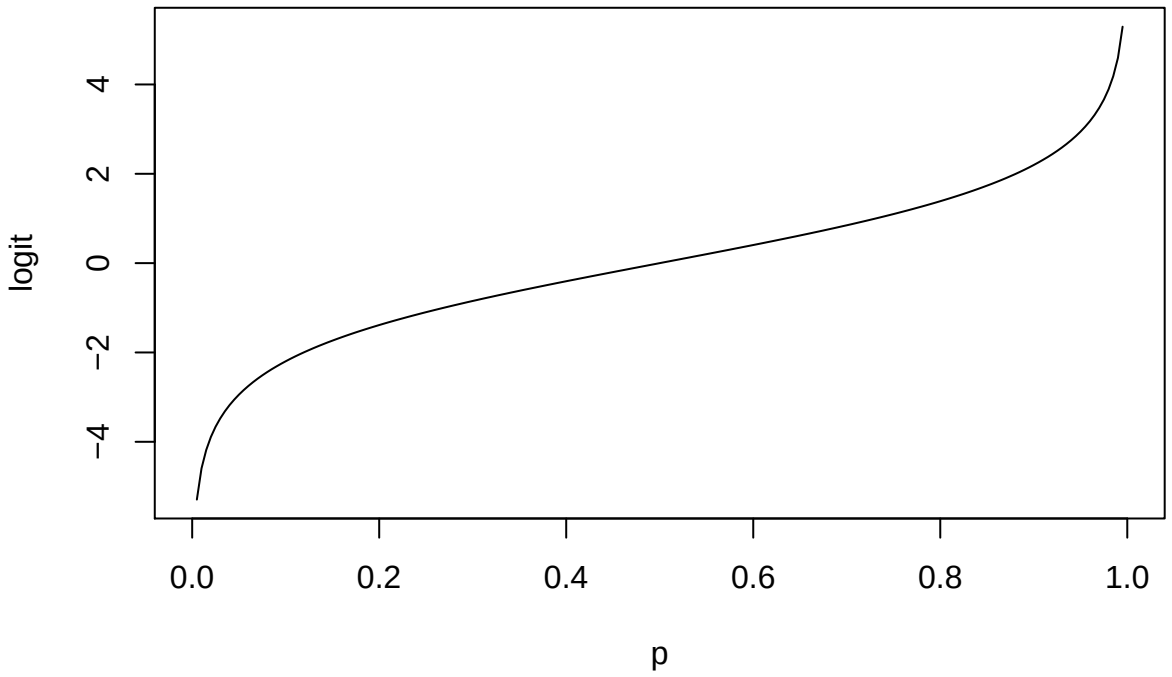
Wtedy

$$log(Odds) = \beta_0 + \beta_1 x_1 + \ldots \beta_k x_k$$

W modelu logitowym, naturalnym sposobem mówienia o losowości jest iloraz szans, a nie p-stwa! Chociaż oczywiście możemy używać p i funkcji logit.

$$logit(p) = \log \frac{p}{1-p} = \beta^T x$$

```
p=seq(0,1, 0.005)
plot(p, log(p/(1-p)), type="l", ylab = "logit")
```



Jeśli znamy β, to dla każdej nowej osoby możemy wyliczyć p-stwo wystąpienia niedokrwiennej choroby serca w przeciągu 10 lat. Wystarczy, że będziemy mieli informacje o zmienych objaśniających.

Pozostaje pytanie jak wyliczyć β. Tutaj z pomocą przychodzi statystyka. Zakładamy, że

$$y \sim B(1, p),$$

gdzie $p = P(y = 1)$. $\hat{\beta}$ wyliczamy przez maksymalizację funkcji wiarogodności

$$l(\beta) = \sum_i \{y_i \cdot \log P(y_i = 1|x_i) + (1 - y_i) \cdot \log P(y_i = 0|x_i)\} = \sum_i y_i \beta^T x - \log(1 + e^{\beta^T x})$$

Policzmy pochodną po β:

$$\frac{\partial l(\beta)}{\partial \beta} = \frac{\partial}{\partial \beta} \sum_i y_i \beta^T x_i - \log(1 + e^{\beta^T x_i}) = \sum_i x_i \left(y_i - \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \ldots \beta_k x_k)}} \right)$$

Niestety mamy do czynienia z nieliniową funkcją od β i nie ma zamkniętej formuły na $\hat{\beta}$. Zamiast tego używa się iteracyjnych algorytmów (np. Newtona–Raphsona).

Po co to wszystko, czyli na jakie pytania może nam dać odpowiedź regresja logistyczna?

Chcemy znaleźć czynniki ryzyka, czyli zmienne, które mają wpływ na występowanie chorób serca.

- BPMeds - czy badany był na lekach na nadciśnienie podczas pierwszego badania
- sysBP - ciśnienie skurczowe
- diaBP - ciśnienie rozkurczowe
- glucose - poziom glukozy (mg/dL)

```
set.seed(23)
split=sample(1:nrow(framingham), round(0.7*nrow(framingham)))
train=framingham[split, ]
test=framingham[-split, ]
```

```r
full_model=glm(TenYearCHD~., data = train, family = "binomial")
summary(full_model)
```

```
##
## Call:
## glm(formula = TenYearCHD ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8843  -0.5915  -0.4358  -0.2984   2.8359
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -7.1516331  0.8468442  -8.445  < 2e-16 ***
## male             0.5002931  0.1316826   3.799 0.000145 ***
## age              0.0534532  0.0080096   6.674  2.5e-11 ***
## education       -0.0477349  0.0593395  -0.804 0.421145
## currentSmoker   -0.2111407  0.1942539  -1.087 0.277067
## cigsPerDay       0.0248318  0.0075668   3.282 0.001032 **
## BPMeds           0.1982018  0.2708402   0.732 0.464289
## prevalentStroke  0.6632421  0.5523656   1.201 0.229856
## prevalentHyp     0.2630118  0.1624786   1.619 0.105502
## diabetes         0.3584420  0.3692186   0.971 0.331642
## totChol          0.0032611  0.0013217   2.467 0.013610 *
## sysBP            0.0188464  0.0045238   4.166  3.1e-05 ***
## diaBP           -0.0135508  0.0076255  -1.777 0.075561 .
## BMI             -0.0009514  0.0149437  -0.064 0.949234
## heartRate       -0.0047571  0.0051537  -0.923 0.355983
## glucose          0.0051933  0.0026367   1.970 0.048878 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2163.5  on 2546  degrees of freedom
## Residual deviance: 1931.1  on 2531  degrees of freedom
##   (421 observations deleted due to missingness)
## AIC: 1963.1
##
## Number of Fisher Scoring iterations: 5
```

Czego nie widzimy powyżej? (w porównaniu z summary dla regresji liniowej)

. . .

. . .

. . .

- R^2
- Testu na istotność względem modelu zerowego (F-test)

Sprawdzanie dokładności modelu na zbiorze testowym

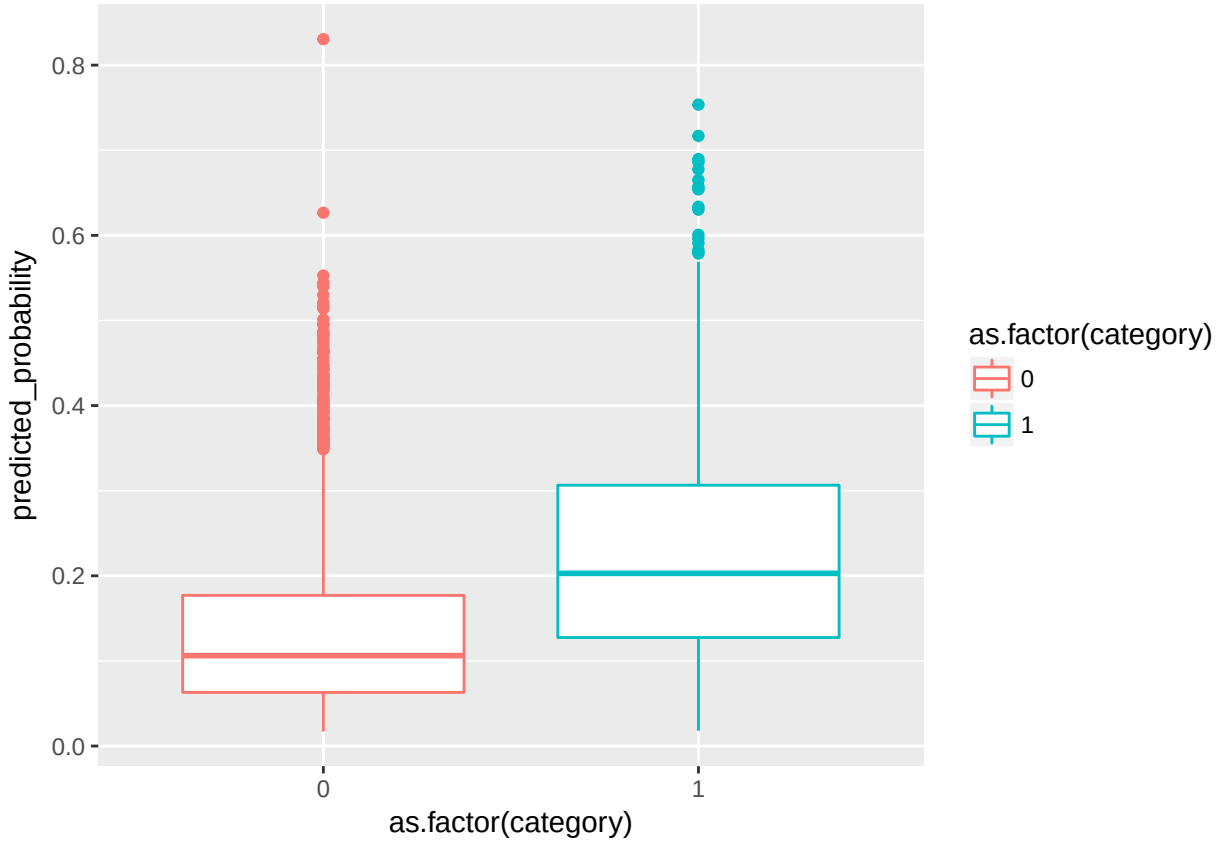
Confusion matrix

Czy nasz model dobrze klasyfikuje obserwacje do kategorii?

```
tapply(full_model$fitted.values, full_model$y, mean)
```

```
##         0         1
## 0.1358239 0.2372696
```

```
ggplot(data.frame(predicted_probability=full_model$fitted.values, category=full_model$y)) +
  geom_boxplot(aes(x=as.factor(category), y=predicted_probability, color=as.factor(category)))
```



Wygląda to w miarę sensownie. Żeby dokonać klasyfikacji musimy przyjąć jakiś punkt odcięcia. Dla jakiego p uznajemy, że $\hat{y} = 1$? Naturalnym wyborem (i należy go stosować, chyba że dobry powód, żeby tego nie robić) jest ustaleniu progu na 0.5. Sprawdźmy jak dobrze klasyfikujemy obserwacje ze zbioru treningowego. Użyjemy do tego Confusion Matrix.

	Przewidziane 0	Przewidziane 1
Faktyczne 0	True Negatives (TN)	False Positives (FP)
Faktyczne 1	False Negatives (FN)	True Positives (TP)

```
knitr::kable( table(full_model$y, full_model$fitted.values>0.5))
```

	FALSE	TRUE
0	2152	10
1	354	31

Czyli poprawnie rozpoznajemy

```
(2152+31)/(2152+31+354+10)
```

```
## [1] 0.8570868
```

przypadków. Ta miara nazywa się dokładnością (accuracy).

Zróbmy teraz predykcję na zbiorze testowym.

```
predictTest=predict(full_model, newdata = test, type = "response")
knitr::kable( table(test$TenYearCHD, predictTest>0.5))
```

	FALSE	TRUE
0	933	6
1	161	11

```
(933+11)/(161+6+933+11)
```

```
## [1] 0.849685
```

Zgodnie z oczekiwaniami predykcja jest nieco gorsza.

Sensitivity, Specificity

Jak inaczej można mierzyć jakość dopasowania modelu?

Jaki procent pozytywnych przypadków nasz model rozponał?

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

Nazywane też True Positive Rate lub Recall.

Jak dużo z obserwacji, które uznaliśmy za negatywne, rzeczywiście są negatywne?

$$\text{Specificity} = \frac{TN}{TN + FP}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

Krzywa ROC

Zauważmy, że miary dobroci predykcji, które rozważaliśmy do tej pory zależą od przyjętego punktu odcięcia dla p. Może nas interesować jakie ogólne własności, niezależne od konkretnego p ma nasz model. W tym celu rozważa się krzywą ROC (Receiver Operator Characteristic).

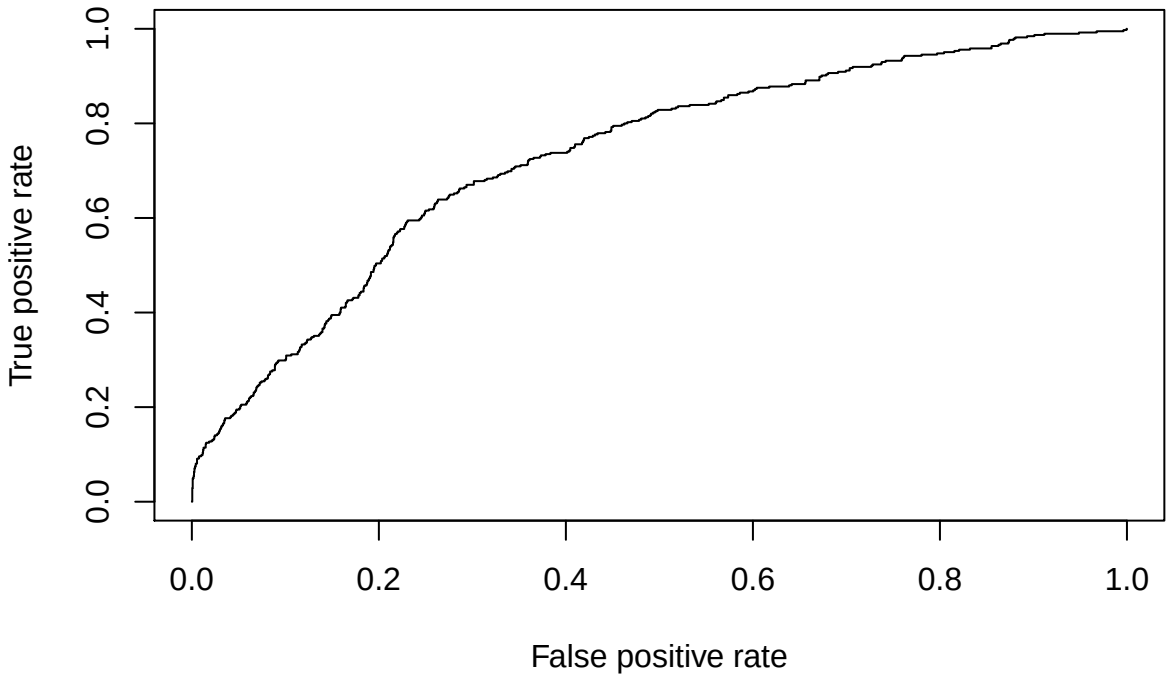
```
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

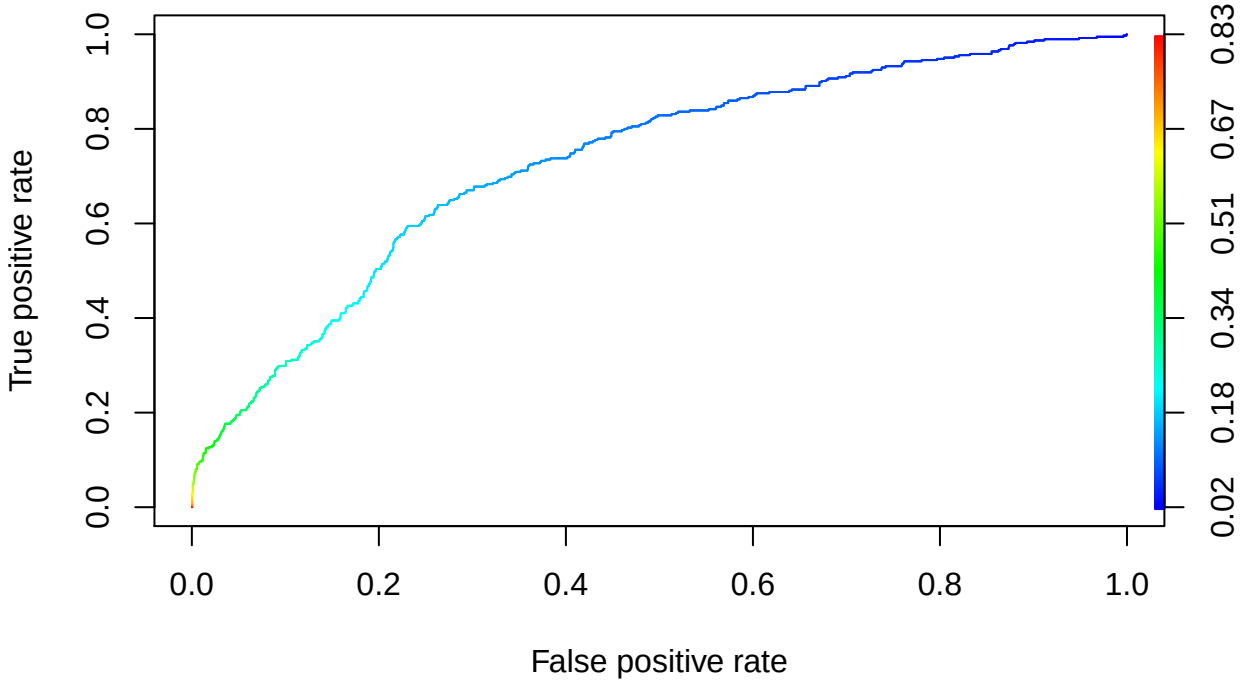
```
ROCRPred=prediction(full_model$fitted.values, full_model$y)
ROCRPerf=performance(ROCRPred, "tpr", "fpr")
plot(ROCRPerf)
```



$$FPR = 1 - \text{Specificity}$$

Ta krzywa jest indesowana p. W jaki sposób?

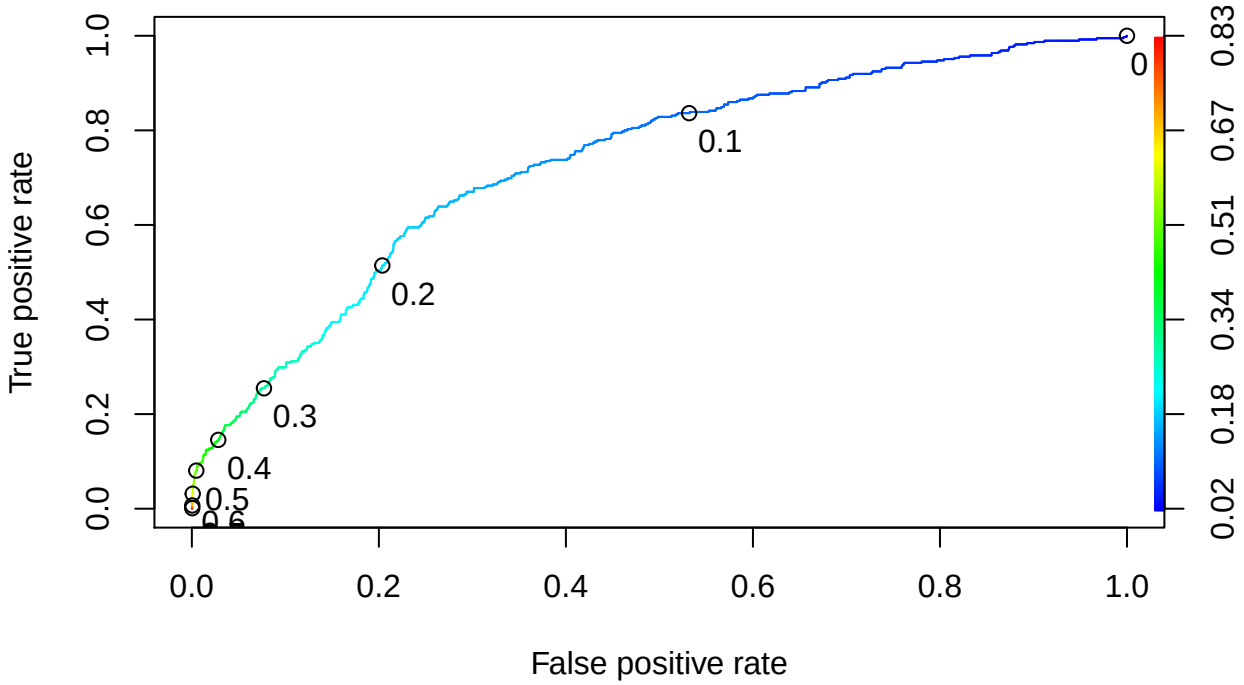
```
plot(ROCRPerf, colorize=TRUE)
```



- Dla $p = 0$ mamy górny prawy róg - wszystkie Negatives są False Positives.
- Dla $p = 1$, mamy lewy dolny róg - wszystkie Positives są False Negatives.

Możemy dodać etykietki

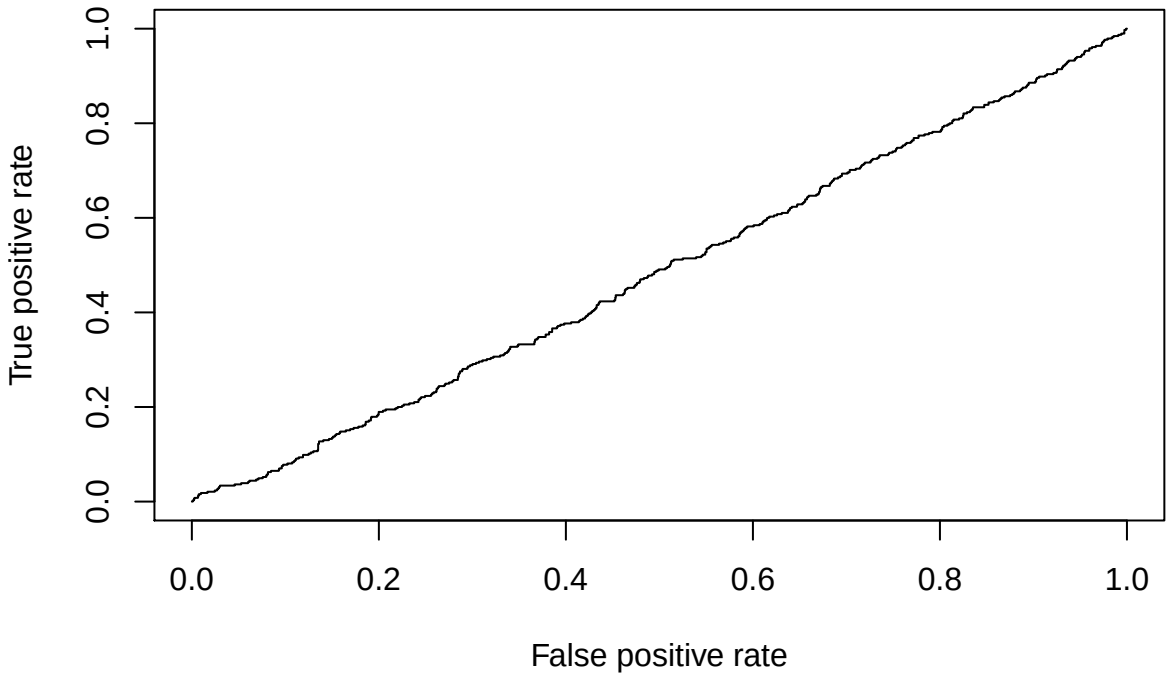
```
plot(ROCRPerf, colorize=TRUE, print.cutoffs.at=seq(0,0.8,0.1), text.adj=c(-0.2, 1.7))
```



AUC

No dobrze, mamy naszą krzywą, ale jak na jej podstawie porównać ze sobą dwa modele? Jak stwierdzić, która krzywa jest lepsza? Zauważmy, ze klasyfikator, który losowo przydzielałby prawdopodobieństwa bez względu na grupy, miałby krzywą AUC, która byłaby, mniej więcej, prostą nachyloną pod kątem 45 stopni.

```
ROCRPred=prediction(runif(length(full_model$y)), full_model$y)
ROCRPerf=performance(ROCRPred, "tpr", "fpr")
plot(ROCRPerf)
```



Zatem miarą dobroci modelu jest oddalenie krzywej od prostej $y = x$, równoważnie pole pod wykresem. AUC jes zatem liczbą z zakresu [0,1]. AUC=0.5 wskazuje na model, który radzi sobie nie lepiej niż losowy predyktor.

Jak duże jest AUC w powyższym przykładzie?

```
auc = as.numeric(performance(ROCRPred, "auc")@y.values)
auc
```

```
## [1] 0.4845225
```

A jak dla modelu?

```
ROCRPred=prediction(full_model$fitted.values, full_model$y)
auc = as.numeric(performance(ROCRPred, "auc")@y.values)
auc
```

```
## [1] 0.7306691
```

A jak na zbiorze testowym?

```
ROCRpredTest = prediction(predictTest, test$TenYearCHD)
auc = as.numeric(performance(ROCRpredTest, "auc")@y.values)
auc
```

```
## [1] 0.7522228
```

Wybór zmiennych

AIC

Popatrzmy jeszcze raz na eRową wyplujkę dla modelu logistycznego

```
summary(full_model)
```

```
##
## Call:
```

```
## glm(formula = TenYearCHD ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8843  -0.5915  -0.4358  -0.2984   2.8359
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -7.1516331  0.8468442  -8.445  < 2e-16 ***
## male             0.5002931  0.1316826   3.799 0.000145 ***
## age              0.0534532  0.0080096   6.674  2.5e-11 ***
## education       -0.0477349  0.0593395  -0.804 0.421145
## currentSmoker   -0.2111407  0.1942539  -1.087 0.277067
## cigsPerDay       0.0248318  0.0075668   3.282 0.001032 **
## BPMeds           0.1982018  0.2708402   0.732 0.464289
## prevalentStroke  0.6632421  0.5523656   1.201 0.229856
## prevalentHyp     0.2630118  0.1624786   1.619 0.105502
## diabetes         0.3584420  0.3692186   0.971 0.331642
## totChol          0.0032611  0.0013217   2.467 0.013610 *
## sysBP            0.0188464  0.0045238   4.166  3.1e-05 ***
## diaBP           -0.0135508  0.0076255  -1.777 0.075561 .
## BMI             -0.0009514  0.0149437  -0.064 0.949234
## heartRate       -0.0047571  0.0051537  -0.923 0.355983
## glucose          0.0051933  0.0026367   1.970 0.048878 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2163.5  on 2546  degrees of freedom
## Residual deviance: 1931.1  on 2531  degrees of freedom
##   (421 observations deleted due to missingness)
## AIC: 1963.1
##
## Number of Fisher Scoring iterations: 5
```

Miarą dobroci dopasowania modelu jest tutaj AIC. Przypomnijmy, że jest to miara, która bierze pod uwagę zarówno dopasowanie do danych, jak i liczbę zmiennych w modelu. Należy pamiętać, że dwa modele można ze sobą porównywać jedynie jeśli są zbudowane na dokładnie tym samym zbiorze obserwacji. AUC jest pod tym względem dużo bardziej czułe niż R^2.

Oczywiście tak jak dla modelu liniowego w R zaimplementowana jest funkcja *step*, która znajduje, po podzbiorach zmiennych, model maksymalizujący AUC.

```
step(full_model)
```

No właśnie, przy budowaniu modeli wyrzucamy obserwacje z brakującymi danymi. Aby móc użyć funckji *step* trzeba albo:

- usunąć obserwacje z brakującymi danymi
- uzupełnić brakujące dane

Regularyzacja

Podobnie jak w przypadku zwykłej regresji możemy chcieć dokonać regresji logistycznej z regularyzacją. Przypomnijmy, że regularyzacji używamy w dwóch przypadkach:

- gdy liczba obserwacji jest porównywalan z liczbą zmiennych
- gdy zależy nam na wyborze zmiennych do modelu

Pakiet **glmnet** nie toleruje wartości NA dlatego:

- albo usuniemy obserwacje zawierające NA
- albo uzupełnimy brakujące dane (więcej np. (tu)[http://stats.stackexchange.com/questions/104194/how-to-handle-with-missing-values-in-order-to-prepare-data-for-feature-selection])

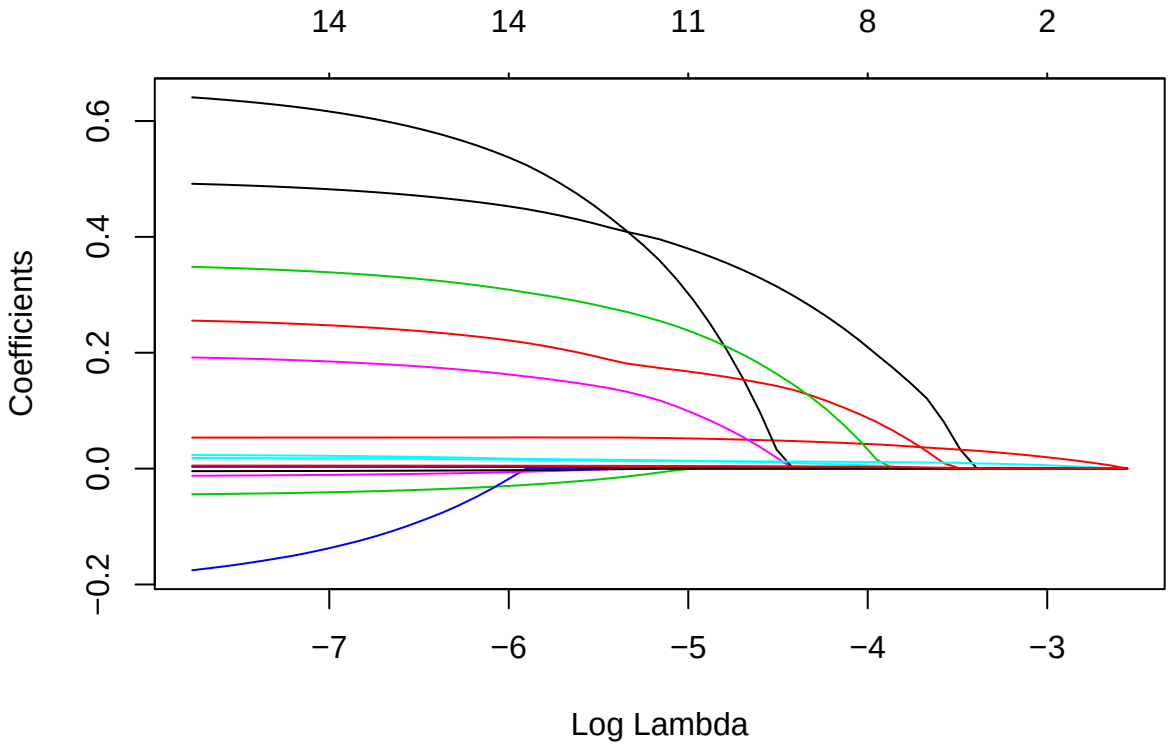
```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-5
```

```
train.glmnet=train[complete.cases(train),]
CHDmodel_lasso=glmnet(x = as.matrix(train.glmnet[,! names(train.glmnet)=='TenYearCHD']), train.glmnet$T
plot(CHDmodel_lasso, xvar = "lambda")
```




```
CHDmodel_lasso=cv.glmnet(x = as.matrix(train.glmnet[,! names(train.glmnet)=='TenYearCHD']), train.glmnet
CHDmodel_lasso$lambda.1se
```

```
## [1] 0.02794979
```

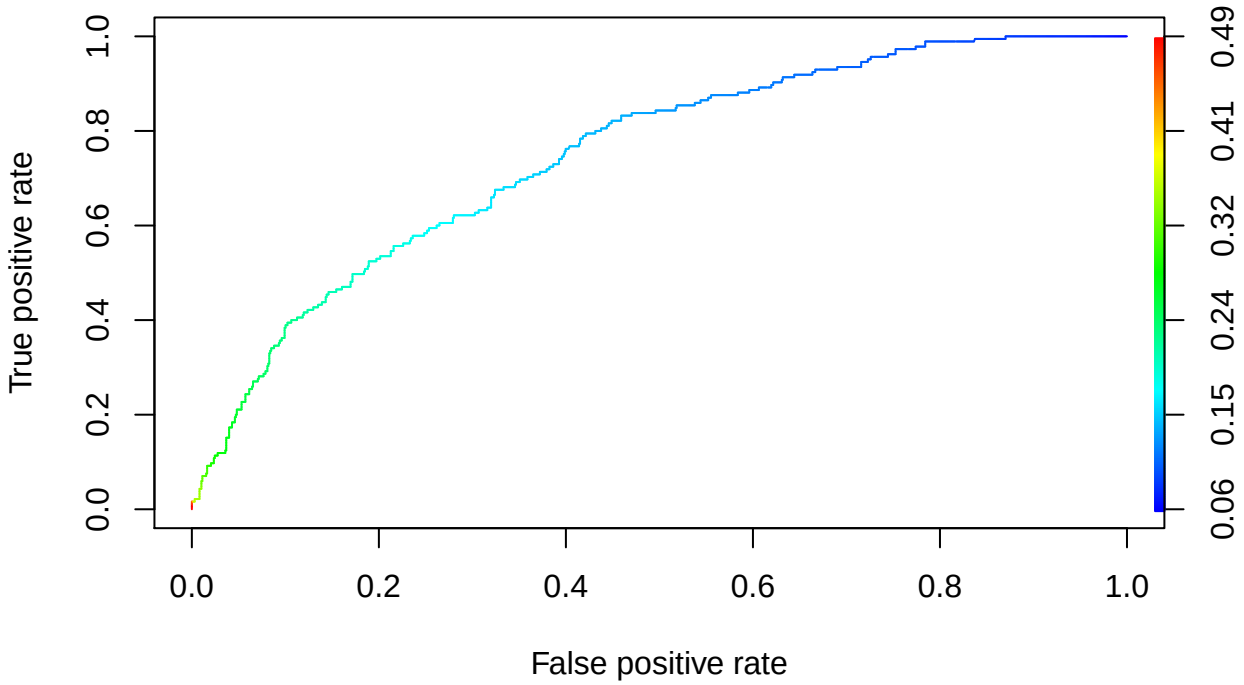
Jak dobrze radzi sobie nasz model?

```
predictTest = predict(CHDmodel_lasso, type="response",
                      newx = as.matrix(test[,! names(test)=='TenYearCHD']),
                      lambda=CHDmodel_lasso$lambda.1se)
knitr::kable( table(test$TenYearCHD, predictTest>0.5))
```

	FALSE
0	978

	FALSE
1	185

```
ROCRpredTest = prediction(predictTest, test$TenYearCHD)
ROCRPerf=performance(ROCRpredTest, "tpr", "fpr")
plot(ROCRPerf, colorize=TRUE)
```



Jak duże jest AUC?

```
auc = as.numeric(performance(ROCRpredTest, "auc")@y.values)
auc
```

```
## [1] 0.7474714
```

Jak wygląda nasz model?

```
which(CHDmodel_lasso$glmnet.fit$lambda==CHDmodel_lasso$lambda.1se)
```

```
## [1] 12
```

```
coeffs=CHDmodel_lasso$glmnet.fit$beta[,which(CHDmodel_lasso$glmnet.fit$lambda==CHDmodel_lasso$lambda.1s
which(coeffs!=0)
```

```
##         male          age prevalentHyp        sysBP      glucose
##            1            2            8           11           15
```