Przewidywanie wynikow meczow za pomoca sieci neuronowej

Jakub Augustynek, Jakub Ignatik, Artur Karamon, Jaroslaw Kmak

17 listopada 2018

Wstęp

Problem przewidywania rezultatu jakim zakończy się mecz piłkarski jest dość złożony. Wpływ na końcowy rezultat ma wiele czynników. Ludzka intuicja podpowiada, iż czynniki takie jak obecna forma poszczególnych zawodników, miejsce rozgrywania spotkania przez daną drużynę (na własnym stadionie, bądź na wyjeździe), czy też obecna lokata w lidze mają istotny wpływ na wynik meczu. Jednakże czasem człowiek, mimo swojej nieświadomości , bierze pod uwagę również takie aspekty jak upodobanie do danej drużyny bądź zawodnika/trenera. Jest to zazwyczaj podejście błędne. W niniejszej pracy do przewidywania końcowego rozstrzygnięcia spotkania wykorzystane zostaną możliwości sztucznych sieci neuronowych. Zaletą takiego rozwiązania są możliwości obliczeniowe obecnych komputerów, które wraz z odpowiednią zaimplementowaną siecią neuronową są w stanie rozwiązywać złożone problemy. Podejście zaproponowane w pracy będzie polegać na wprowadzeniu do sieci neuronowej czynników ukazujących osiągnięcia danego zespołu w poprzednich meczach, by następnie na podstawie tych danych sieć wykryła zależności łączące je z końcowym wynikiem spotkania.
Dane: Do projektu wykorzystaliśmy dane dla angielskiej Premier League (http://www.football-data.co.uk/englandm.php), a konkretnie sezon 2017/2018. W tym sezonie rozgrywkę prowadziło ze sobą 20 klubów piłkarskich. Dla każdego starcia między nimi baza zawiera statystyki meczowe w postaci m.in. strzelonych goli, straconych bramek, liczby rzutów rożnych czy też ilości spalonych.
Biblioteka: Projekt wykorzystuje bibliotekę “neuralnet”, dostępną w programie R. Pakiet ten umożliwia zbudowanie sieci neuronowej i jej uczenie.

Model

Model, który ostatecznie wybraliśmy prentuje się następująco:

```
## [1] "model_ssn<- neuralnet(FTR ~ WinRatioHome + WinRatioAway + PositionHome + PositionAway, datatrain , h
idden = c(5,2,2),linear.output = T )"
```

Zmienne niezależne:
-WinRatioHome - % wygranych meczy przez gospodarza
-WInRatioAway - % wygranych meczy przez gościa
-PositionHome - pozycja gospodarza w rankingu
-PositionAway - pozycja gościa w rankingu
Zmienna zależna:
-FTR - wynik meczu (1 - wygrana gospodarza, 0 - remis, -1 - wygrana gościa)
Ukryte warstwy: (5) (2) (2)

Powyższy model jest tym, który osiągnął najwyższą skuteczność ze wszystkich modeli przez nas testowanych (0,5). Próbowaliśmy m.in. jako zmienne niezależne wstawić gole zdobyte i stracone w paru ostatnich meczach czy też punkty zespołu ze strony FIFA.
Poniżej znajduje się kod, w którym utworzyliśmy oraz przetestowaliśmy sieć neuronową. W naszym rozwiązaniu komputer sam dobiera liczbę neuronów oraz warstw ukrytych, jednak ograniczyliśmy z góry tą liczebność, gdyż moc obliczeniowa komputera nie pozwalała na przetestowanie tego na wyższej liczbie warstw i neuronów. Należy jednak zauważyć, że znaleziona przez nas liczba warstw i neuronów jest dosyć mała, można więc założyć, że powiększenie zbioru do przeszukania nie poprawiłoby znacząco skuteczności naszego modelu.

```
mecze_PL<-NULL
mecze<-NULL

#wczytanie bazy danych zawierającej wyniki meczów wszystkich kolejek Premier League w latach 2017/2018
mecze_PL <- read.csv("C:/Users/Jan/Desktop/PL_mecze_17_18.csv", header=T)

############################### PRZYGOTOWANIE DANYCH ################################
mecze_PL<-mecze_PL[,2:10]
mecze_PL$Date<-rev(mecze_PL$Date)
mecze_PL$HomeTeam<-rev(mecze_PL$HomeTeam)
mecze_PL$AwayTeam<-rev(mecze_PL$AwayTeam)
mecze_PL$FTHG<-rev(mecze_PL$FTHG)
mecze_PL$FTAG<-rev(mecze_PL$FTAG)
mecze_PL$FTR<-rev(mecze_PL$FTR)
mecze_PL$HTHG<-rev(mecze_PL$HTHG)
mecze_PL$HTAG<-rev(mecze_PL$HTAG)
mecze_PL$HTR<-rev(mecze_PL$HTR)
mecze_PL$FTR<-as.numeric(mecze_PL$FTR)
mecze_PL$FTR[mecze_PL$FTR==1]<--1
mecze_PL$FTR[mecze_PL$FTR==2]<-0
```

```
mecze_PL$FTR[mecze_PL$FTR==3]<-1

position_df<-data.frame(team=c("Chelsea","Tottenham", "Man City", "Liverpool", "Arsenal", "Man United","Ever
ton","Southampton","Bournemouth","West Brom", "West Ham", "Leicester","Stoke","Crystal Palace", "Swansea", "
Burnley", "Watford", "Brighton", "Newcastle","Huddersfield"), position=c(1:17,rep(21,3)))

mecze<-mecze_PL
mecze_PL2<-mecze_PL
mecze_PL$WinRatioHome<-rep(0,nrow(mecze_PL))
mecze_PL$WinRatioAway<-rep(0,nrow(mecze_PL))
mecze_PL$LostRatioHome<-rep(0,nrow(mecze_PL))
mecze_PL$LostRatioAway<-rep(0,nrow(mecze_PL))
zespoly<-unique(mecze_PL$HomeTeam)

for (zespol in zespoly) {

  #wybór drużyny, dla której ma zostać utworzona sieć neuronowa
  team<-zespol
  mecze<-mecze_PL2

  #przekształcenie ramki danych tak, aby zawierała one jedynie mecze, w których brał    udział wybrany klub
  mecze<-mecze[mecze$HomeTeam==team|mecze$AwayTeam==team,]
  mecze$IsHomeTeam<-rep(0,nrow(mecze))
  mecze$IsHomeTeam[mecze$HomeTeam==team]<-1
  mecze$Scored[mecze$HomeTeam==team]<-mecze$FTHG[mecze$HomeTeam==team]
  mecze$Scored[mecze$HomeTeam!=team]<-mecze$FTAG[mecze$HomeTeam!=team]
  mecze$Lost[mecze$HomeTeam!=team]<-mecze$FTHG[mecze$HomeTeam!=team]
  mecze$Lost[mecze$HomeTeam==team]<-mecze$FTAG[mecze$HomeTeam==team]

  mecze<-mecze[,-c(4:5)]
  w_Scored<-mecze$Scored
  w_Lost<-mecze$Lost

  #dodanie kolumny wyników
  WL_vector<-sapply(mecze$Scored-mecze$Lost, function(x){
    if(x>0){res<-1}
    if(x<0){res<--1}
    if(x==0){res<-0}
    res})
  mecze$Result<-WL_vector
  w_Result<-mecze$Result
  mecze<-mecze[-c((length(mecze$Scored)-3):length(mecze$Scored)),]

  #utworzenie dodatkowych kolumn, reprezentujących poprzednie mecze
  mecze$Scored1<-w_Scored[2:(length(w_Scored)-3)]
  mecze$Lost1<-w_Lost[2:(length(w_Lost)-3)]
  mecze$Result1<-w_Result[2:(length(w_Result)-3)]

  mecze$Scored2<-w_Scored[3:(length(w_Scored)-2)]
  mecze$Lost2<-w_Lost[3:(length(w_Lost)-2)]
  mecze$Result2<-w_Result[3:(length(w_Result)-2)]

  mecze$Scored3<-w_Scored[4:(length(w_Scored)-1)]
  mecze$Lost3<-w_Lost[4:(length(w_Lost)-1)]
  mecze$Result3<-w_Result[4:(length(w_Result)-1)]

  mecze$Scored4<-w_Scored[5:(length(w_Scored))]
  mecze$Lost4<-w_Lost[5:(length(w_Lost))]
  mecze$Result4<-w_Result[5:(length(w_Result))]

  mecze$WinRatio<-((mecze$Result1==1)+(mecze$Result2==1)+(mecze$Result3==1)+(mecze$Result4==1))/4
  mecze$LostRatio<-((mecze$Result1==-1)+(mecze$Result2==-1)+(mecze$Result3==-1)+(mecze$Result4==-1))/4

  mecze_PL$WinRatioHome[mecze_PL$HomeTeam==zespol][1:length(mecze$WinRatio[mecze$IsHomeTeam==1])]<-mecze$Win
Ratio[mecze$IsHomeTeam==1]
  mecze_PL$WinRatioAway[mecze_PL$AwayTeam==zespol][1:length(mecze$WinRatio[mecze$IsHomeTeam==0])]<-mecze$Win
Ratio[mecze$IsHomeTeam==0]
  mecze_PL$LostRatioHome[mecze_PL$HomeTeam==zespol][1:length(mecze$LostRatio[mecze$IsHomeTeam==1])]<-mecze$L
ostRatio[mecze$IsHomeTeam==1]
  mecze_PL$LostRatioAway[mecze_PL$AwayTeam==zespol][1:length(mecze$LostRatio[mecze$IsHomeTeam==0])]<-mecze$L
ostRatio[mecze$IsHomeTeam==0]
  mecze_PL$PositionHome[mecze_PL$HomeTeam==zespol]<-position_df$position[position_df$team==zespol]
```

```
  mecze_PL$PositionAway[mecze_PL$AwayTeam==zespol]<-position_df$position[position_df$team==zespol]
}

mecze_PL<-mecze_PL[1:(nrow(mecze_PL)-40),]
mecze<-mecze_PL

#wylosowanie próby zawierającej indeksy wektora treningowego (70% zbioru)
set.seed(123)
dl_w_train <- 0.70 * nrow(mecze)
index <- sample( seq_len ( nrow ( mecze ) ), size = dl_w_train )

#skalowanie danych
maxs <- apply(mecze[,c(4:6,10:15)], 2, max)
mins <- apply(mecze[,c(4:6,10:15)], 2, min)
mecze_scaled <- as.data.frame(scale(mecze[,c(4:6,10:15)], center = mins, scale = maxs - mins))
mecze[,match(colnames(mecze_scaled),colnames(mecze))]<-mecze_scaled

# podział na dane treningowe i testowe
datatrain <- mecze[ index, ]
datatest <- mecze[ -index, ]

########################### PREDYKCJA I DOSTOSOWANIE MODELU ########################

#dostępne dla pętli liczby neuronów w każdej z (maksymalnie) trzech warstw
v_1<-4:5
v_2<-1:2
v_3<-1:2

acc_max<-0

iter<-0
for (i1 in v_1) {
  for (i2 in v_2) {
    for (i3 in v_3) {
      iter<-iter+1

      #dopasowanie modelu sieci neuronowej - model FTR, ostateczny
      set.seed(123)
      model_WL<-tryCatch({
        model_WL<- neuralnet(FTR ~ WinRatioHome + WinRatioAway + PositionHome + PositionAway, datatrain , hi
dden = c(i1,i2,i3),linear.output = T )
      },
      error = function(cond){
        return (model_WL<- neuralnet(FTR ~ WinRatioHome + WinRatioAway + PositionHome + PositionAway, datatr
ain, linear.output = T )
        )},
      warning=function(cond){
        return (model_WL<- neuralnet(FTR ~ WinRatioHome + WinRatioAway + PositionHome + PositionAway, datatr
ain,linear.output = T )
        )},
      finally = function(cond){
        return (model_WL<- neuralnet(FTR ~ WinRatioHome + WinRatioAway + PositionHome + PositionAway, datatr
ain , linear.output = T )
        )}
      )

      ## predykcja z wykorzystaniem modelu sieci neuronowej - model FTR
      predict_testWL <- neuralnet::compute(model_WL, datatest[,c(10:11,14:15)])

      #datatest$FTR
      predict_rescaled <- predict_testWL$net.result*(max(mecze_PL$FTR)-min(mecze_PL$FTR))+min(mecze_PL$FTR)
      FTR_rescaled <- (datatest$FTR)*(max(mecze_PL$FTR)-min(mecze_PL$FTR))+min(mecze_PL$FTR)

      predicted_FTR<-as.vector(round(predict_rescaled))
      conf_matrix_FTR<-table(predicted_FTR ,FTR_rescaled)
      #dokładność predykcji goli strzelonych
      accuracy_FTR<-sum(diag(conf_matrix_FTR))/sum(conf_matrix_FTR)

      #wybieranie optymalnej liczby warstw, neuronów oraz skuteczności predykcji
      #obliczanie błędu
      if (accuracy_FTR>acc_max) {
        acc_max<-accuracy_FTR
        v_max<-c(i1,i2,i3)
```

```
        MSE.nn <- sum((predict_rescaled - FTR_rescaled)^2)/nrow(datatest)
      }
    }
  }
}
print(acc_max)
```

```
## [1] 0.5048543689
```

```
print(v_max)
```

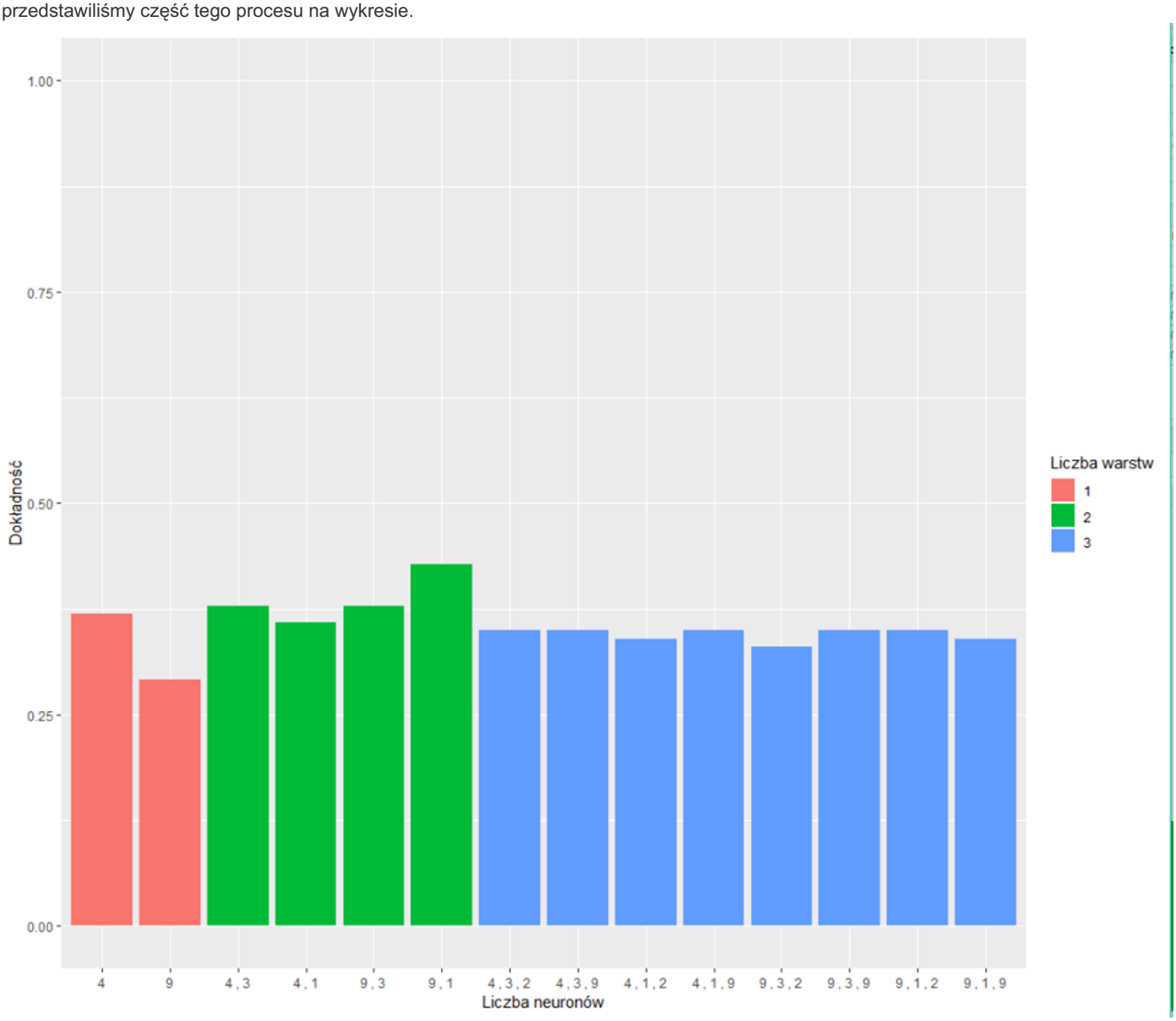
```
## [1] 5 2 2
```

```
print(MSE.nn)
```

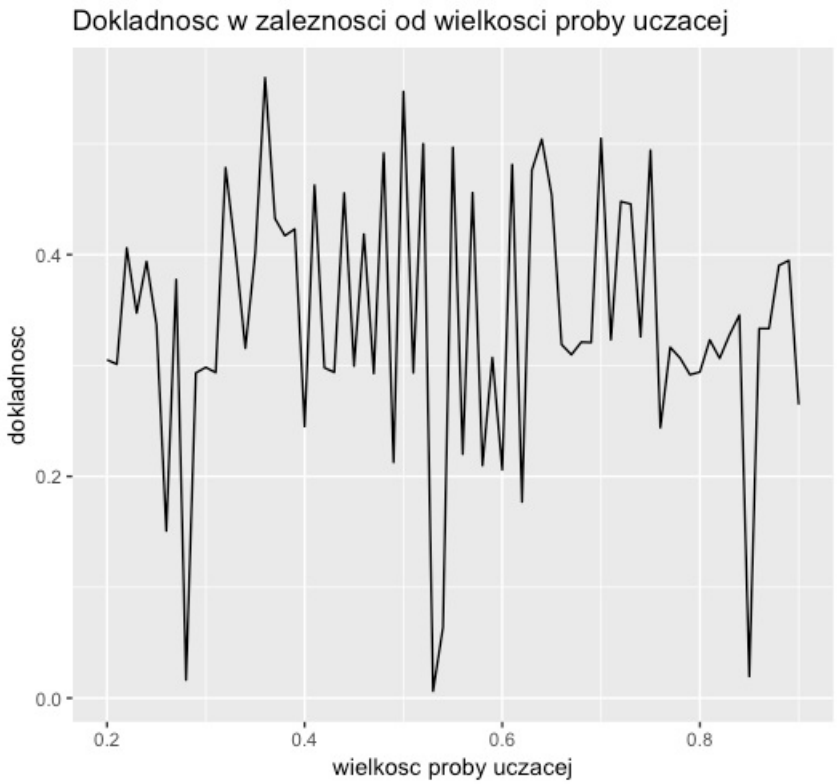
```
## [1] 0.9649255533
```

Zmiana liczby neuronów i warstw

Ponizej znajduje się wykres przedstawiający celność predykcji względem liczby warstw i neuronów. Zrobiła to funkcja, ale na wykresie przedstawiliśmy część tego procesu na wykresie.



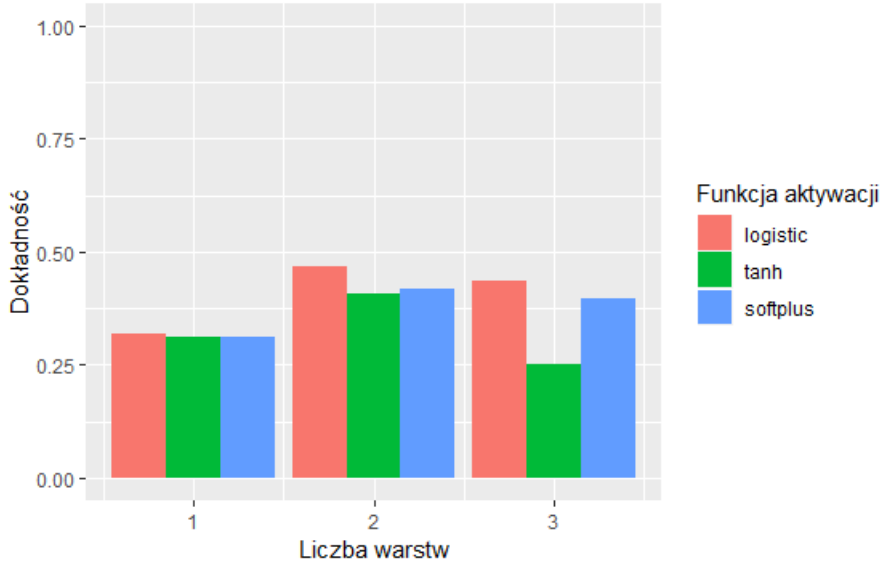
Jak widać, nasza liczba warstw ukrytych i neuronów jest lepsza niż wszystkie pokazane na wykresie.
##Zmiana proporcji zbioru uczącego i testowego
Poniżej znajduje się wykres, który pokaże jak zmieniała się dokładność predykcji wraz ze zmianą proporcji zbioru uczącego i testowego.



Jak widać, dla wielkości zbioru uczącego równej 0,7 nie jest osiągany najlepszy wynik, możnaby go podwyższyć poprzez obniżenie tej wartości do 0,5 lub 0,35, ale wtedy model byłby przeuczony i pojawienie się nowych danych spowodowałoby obniżenie celności predykcji.

Funkcja aktywacji

Do analizy dokładności predykcji wybraliśmy trzy funkcje aktywacji: logistic (użyta w projekcie), tanh (hiperboliczna) oraz softplus (x^+).



Można zauważyć, że dla wybranej przez nas liczby warstw najskuteczniejszą jest zawsze domyślna funkcja aktywacji, czyli logistyczna. Najgorzej wypada funkcja hiperboliczna, pogarszając swój wynik z każdą nową warstwą.

Model ekonometryczny

Wykonany przez nas model to model logitowy wielomianowy, który przyjmuje 3 wartości: 0 (wygrana gościa), 0,5 (remis) oraz 1 (wygrana gospodarza), zatem podobnie jak przy sieci neuronowej.

```
#utworzenie modelu
model_Log<-multinom(FTR ~ WinRatioHome + WinRatioAway + PositionHome + PositionAway, datatrain)
```

```
## # weights:  18 (10 variable)
## initial  value 260.371112
## iter  10 value 227.675843
## final  value 227.672621
## converged
```

```
model_Log
```

```
## Call:
## multinom(formula = FTR ~ WinRatioHome + WinRatioAway + PositionHome +
##     PositionAway, data = datatrain)
##
## Coefficients:
##        (Intercept) WinRatioHome WinRatioAway  PositionHome PositionAway
## 0.5 -0.09744278084 0.2615066290 -1.015111748 -0.4069079922  1.573085816
## 1    1.35378941379 0.5497024434 -1.907912071 -2.0676854962  1.702823844
##
## Residual Deviance: 455.345242
## AIC: 475.345242
```

```
#policzenie skuteczności predykcji oraz błędu
predict_Log<-predict(model_Log,datatest[,c(10:11,14:15)])
conf_matrix_Log<-table(datatest$FTR,predict_Log)
accuracy_Log<-sum(diag(conf_matrix_Log))/sum(conf_matrix_Log)
predict_Log<-as.numeric(predict_Log)
MSE.multinom <- sum((predict_Log - datatest$FTR)^2)/nrow(datatest)
accuracy_Log
```

```
## [1] 0.4854368932
```

```
MSE.multinom
```

```
## [1] 3.961165049
```

Jak wynika z powyższego kodu, model posiada skuteczność równą 0.4854368932, czyli niewiele mniejszą niż utworzona przez nas sieć neuronowa.
Kolejnym modelem będzie model liniowy, który przyjmuje wartości od 0 do 1, gdzie 0 to wygrana gościa, a 1 to wygrana gospodarza.

```
#utworzenie modelu (wyeliminowana została zmienna WinRatioHome, gdyż nie była istotna statystycznie)

modelEkon<-lm(FTR~WinRatioAway+PositionHome+PositionAway,datatrain)
modelEkon
```

```
##
## Call:
## lm(formula = FTR ~ WinRatioAway + PositionHome + PositionAway,
##     data = datatrain)
##
## Coefficients:
##  (Intercept)  WinRatioAway  PositionHome  PositionAway
##    0.7627379    -0.2986093    -0.3452389     0.2317273
```

```
#policzenie skuteczności predykcji

predict_ekon<-predict(modelEkon,datatest[,c(11,14:15)])
predicted_ekon<-as.vector(round(predict_ekon))
conf_matrix_ekon<-table(datatest$FTR,predicted_ekon)
accuracy_ekon<-sum(diag(conf_matrix_ekon))/sum(conf_matrix_ekon)
MSE.lm <- sum((predict_ekon - datatest$FTR)^2)/nrow(datatest)
accuracy_ekon
```

```
## [1] 0.359223301
```

```
MSE.lm
```

```
## [1] 0.1474410538
```

Model liniowy ma skuteczność na poziomie 36%, jest więc najgorszym z wszystkich przetestowanych modeli, a w dodatku jego interpretacja również stawia wiele zarzutów. Jako jedyny nie wykorzystuje on sztucznych sieci neuronowych (model wielomianowy logitowy je wykorzystywał).

Podsumowanie

Nasza sieć okazała się w przewidywaniu wyników niewiele gorsza od popularnych serwisów bukmacherskich (52%-53%). Trzeba też zwrócić uwagę na błąd średniokwadratowy, który nie był jednak najniższy dla sieci neuronowej (0.96) - dla modeli logitowego i liniowego wynosił on odpowiednio 3.96 i 0.15, więc model liniowy okazał się pod tym względem lepszy.
Należy brać poprawkę na to, że nasza sieć przewiduje wyłącznie wyniki w Premier League, jednak jest to, naszym zdaniem, wystarczająco wysoki poziom, aby nazwać nasz projekt udanym.

Literatura:

Wprowadzenie do tematyki sieci neuronowych. Poznanie zasad funkcjonowania i sposobów wykorzystania:
K. Gurney, An introduction to neural networks, UCL Press, London and New York 1997
D. Kriesel, A Brief Introduction to Neural Networks, Bonn 2005
R. Tadeusiewicz, M. Szaleniec, Leksykon Sieci Neuronowych, Wydawnictwo Fundacji „Projekt Nauka”, Wrocław 2015
Poznanie wyników innych prac dotyczących poruszanego tematu:
www.andrew.carterlunn.co.uk/programming/2018/02/20/beating-the-bookmakers-with-tensorflow.html?fbclid=IwAR00vVVUwUMDl0fr_aQSvWVuY1-UNHw5K1JCIqSzCh2YC-HBL3WpVP2uUFg
Ogólne rozeznanie w temacie sieci neuronowych, poznanie dostępnych możliwości konstrukcji i modyfikacji sieci, tj. funkcje aktywacji:
www.en.wikipedia.org/wiki/Artificial_neural_network
www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html
Wykorzystanie sieci neuronowych w programie RStudio:
www.r-bloggers.com/fitting-a-neural-network-in-r-neuralnet-package/