MATH 449 Final Project

Erika Iwule

May 18th, 2024

A Logistic Regression Project

NBA Injury data set

y = Likelihood of Injury

6 Explanatory Variables: Player's Age, Player's Weight, Player's Height, Previous Injuries, Training Intensity, ans Recovery Time.

1000 observations

```r
library(readr)
injury <- read_csv("injury_data (1).csv")
```

```
## Rows: 1000 Columns: 7
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (7): Player_Age, Player_Weight, Player_Height, Previous_Injuries, Traini...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
injury
```

```
## # A tibble: 1,000 × 7
##    Player_Age Player_Weight Player_Height Previous_Injuries Training_Intensity
##         <dbl>         <dbl>         <dbl>             <dbl>              <dbl>
##  1         24          66.3          176.                 1              0.458
##  2         37          71.0          175.                 0              0.227
##  3         32          80.1          186.                 0              0.614
##  4         28          87.5          176.                 1              0.253
##  5         25          84.7          190.                 0              0.578
##  6         38          75.8          207.                 1              0.359
##  7         24          70.1          177.                 0              0.824
##  8         36          79.0          182.                 1              0.821
##  9         28          64.1          184.                 1              0.477
## 10         28          66.8          198.                 1              0.351
## # ℹ 990 more rows
## # ℹ 2 more variables: Recovery_Time <dbl>, Likelihood_of_Injury <dbl>
```

2. Fit a logistic regression model with all predictors

```r
g1 = glm(Likelihood_of_Injury ~ Player_Age + Player_Weight + Player_Height + Previous_Injuries + Training_Intensi
ty + Recovery_Time, family=binomial, data=injury)
(g2=summary(g1))
```

```
##
## Call:
## glm(formula = Likelihood_of_Injury ~ Player_Age + Player_Weight +
##     Player_Height + Previous_Injuries + Training_Intensity +
##     Recovery_Time, family = binomial, data = injury)
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -1.171427   1.273131  -0.920  0.35751
## Player_Age         -0.001084   0.009772  -0.111  0.91165
## Player_Weight      -0.001053   0.006462  -0.163  0.87061
## Player_Height       0.005254   0.006453   0.814  0.41550
## Previous_Injuries   0.160139   0.127466   1.256  0.20900
## Training_Intensity  0.625049   0.223865   2.792  0.00524 **
## Recovery_Time      -0.015224   0.037536  -0.406  0.68505
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1386.3  on 999  degrees of freedom
## Residual deviance: 1375.9  on 993  degrees of freedom
## AIC: 1389.9
##
## Number of Fisher Scoring iterations: 3
```

Training Intensity is the only significant variable.

3a. Select the best subset of variables.

```r
step(g1)
```

```
## Start:  AIC=1389.88
## Likelihood_of_Injury ~ Player_Age + Player_Weight + Player_Height + 
##     Previous_Injuries + Training_Intensity + Recovery_Time
## 
##                      Df Deviance    AIC
## - Player_Age          1   1375.9 1387.9
## - Player_Weight       1   1375.9 1387.9
## - Recovery_Time       1   1376.0 1388.0
## - Player_Height       1   1376.5 1388.5
## - Previous_Injuries   1   1377.5 1389.5
## <none>                    1375.9 1389.9
## - Training_Intensity  1   1383.7 1395.7
## 
## Step:  AIC=1387.89
## Likelihood_of_Injury ~ Player_Weight + Player_Height + Previous_Injuries + 
##     Training_Intensity + Recovery_Time
## 
##                      Df Deviance    AIC
## - Player_Weight       1   1375.9 1385.9
## - Recovery_Time       1   1376.0 1386.0
## - Player_Height       1   1376.5 1386.5
## - Previous_Injuries   1   1377.5 1387.5
## <none>                    1375.9 1387.9
## - Training_Intensity  1   1383.7 1393.7
## 
## Step:  AIC=1385.92
## Likelihood_of_Injury ~ Player_Height + Previous_Injuries + Training_Intensity + 
##     Recovery_Time
## 
##                      Df Deviance    AIC
## - Recovery_Time       1   1376.1 1384.1
## - Player_Height       1   1376.6 1384.6
## - Previous_Injuries   1   1377.5 1385.5
## <none>                    1375.9 1385.9
## - Training_Intensity  1   1383.7 1391.7
## 
## Step:  AIC=1384.07
## Likelihood_of_Injury ~ Player_Height + Previous_Injuries + Training_Intensity
## 
##                      Df Deviance    AIC
## - Player_Height       1   1376.7 1382.7
## - Previous_Injuries   1   1377.7 1383.7
## <none>                    1376.1 1384.1
## - Training_Intensity  1   1384.0 1390.0
## 
## Step:  AIC=1382.7
## Likelihood_of_Injury ~ Previous_Injuries + Training_Intensity
## 
##                      Df Deviance    AIC
## - Previous_Injuries   1   1378.3 1382.3
## <none>                    1376.7 1382.7
## - Training_Intensity  1   1384.8 1388.8
## 
## Step:  AIC=1382.31
## Likelihood_of_Injury ~ Training_Intensity
## 
##                      Df Deviance    AIC
## <none>                    1378.3 1382.3
## - Training_Intensity  1   1386.3 1388.3
```

```
## 
## Call:  glm(formula = Likelihood_of_Injury ~ Training_Intensity, family = binomial, 
##     data = injury)
## 
## Coefficients:
##        (Intercept)  Training_Intensity  
##            -0.3076              0.6271  
## 
## Degrees of Freedom: 999 Total (i.e. Null);  998 Residual
## Null Deviance:       1386 
## Residual Deviance: 1378  AIC: 1382
```

The best model with the lowest AIC is the model with Training_Intensity as the only explanatory variable.

pi_hat = -0.3076 + 0.6271x

3b. Perform a diagnostic on the best model.

```
g3 = glm(formula = Likelihood_of_Injury ~ Training_Intensity, family = binomial, 
    data = injury)
g4 = summary(g3)
g4
```

```
## 
## Call:
## glm(formula = Likelihood_of_Injury ~ Training_Intensity, family = binomial, 
##     data = injury)
## 
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)   
## (Intercept)         -0.3076     0.1264  -2.434  0.01492 * 
## Training_Intensity   0.6271     0.2227   2.815  0.00487 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 1386.3  on 999  degrees of freedom
## Residual deviance: 1378.3  on 998  degrees of freedom
## AIC: 1382.3
## 
## Number of Fisher Scoring iterations: 3
```

3c. Perform all possible inferences you can think about.

Wald Test and Wald CI

```
beta=g4$coef[2,1]
alpha=0.05
z_a=qnorm(1-alpha/2)
se=g4$coef[2,2]

c(beta-z_a*se, beta+z_a*se)
```

```
## [1] 0.1905512 1.0636297
```

H0: B = 0

H1: B does not equal 0

Z-value = 2.815

P-value = 0.00487 < 0.05

We reject HO. There is significant evidence that training intensity is not independent from the likelihood of injury for NBA Players.

95% Wald CI is (0.1905512, 1.0636297). We are 95% confident that beta is between (0.1905512, 1.0636297).

Likelihood Ratio Test and CI

```
drop1(g3, test="Chisq")
```

```
## Single term deletions
##
## Model:
## Likelihood_of_Injury ~ Training_Intensity
##                    Df Deviance    AIC    LRT Pr(>Chi)
## <none>                  1378.3 1382.3
## Training_Intensity  1   1386.3 1388.3 7.9851 0.004716 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
confint(g3)
```

```
## Waiting for profiling to be done...
```

```
##                         2.5 %      97.5 %
## (Intercept)        -0.5562336 -0.06059448
## Training_Intensity  0.1917668  1.06537200
```

H0: B = 0

H1: B does not equal 0

LRT = 7.9851 df = 1

P-value = 0.004716 < 0.05

We reject HO. There is significant evidence that training intensity is not independent from the likelihood of injury for NBA Players.

95% LRT CI is (0.1917668, 1.06537200). We are 95% confident that beta is between (0.1917668, 1.06537200).

Marginal Effect

```
library(mfx)
```

```
## Loading required package: sandwich
```

```
## Loading required package: lmtest
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: MASS
```

```
## Loading required package: betareg
```

```
logitmfx(g3, atmean=FALSE, data=injury)
```

```
## Call:
## logitmfx(formula = g3, data = injury, atmean = FALSE)
##
## Marginal Effects:
##                       dF/dx Std. Err.      z    P>|z|
## Training_Intensity 0.155524  0.056117 2.7714 0.005581 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The average marginal effect is 0.022584.

Wald test for the training intensity effect and a 95% Wald confidence interval for the odds ratio corresponding to a 1-unit increase training effect

```
g4
```

```
##
## Call:
## glm(formula = Likelihood_of_Injury ~ Training_Intensity, family = binomial,
##     data = injury)
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -0.3076     0.1264  -2.434  0.01492 *
## Training_Intensity   0.6271     0.2227   2.815  0.00487 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1386.3  on 999  degrees of freedom
## Residual deviance: 1378.3  on 998  degrees of freedom
## AIC: 1382.3
##
## Number of Fisher Scoring iterations: 3
```

```
z = g4$coef[2]/0.2227
z2 = z^2
z2
```

```
## [1] 7.929028
```

```
c1 = g4$coef[2] - (1.96)*(0.2227)
c2 = g4$coef[2] + (1.96)*(0.2227)
exp(c1)
```

```
## [1] 1.209973
```

```
exp(c2)
```

```
## [1] 2.89673
```

Wald Test

H0: B = 0

H1: B does not equal 0

z= 2.815

z^2= 7.929028 df=1

p-value = 0.00487 < 0.05

We reject H0. There is significant evidence that beta does not equal 0.

The 95% Wald CI is (1.209973, 2.89673). We are 95% confident that the the odds ratio corresponding to a 1-unit increase in training intensity has at least 20.9% increase and at most a 289% increase in odds of likelihood of injury.

likelihood-ratio test and construct a 95% profile likelihood interval

```
drop1(g3, test="Chisq")
```

```
## Single term deletions
##
## Model:
## Likelihood_of_Injury ~ Training_Intensity
##                    Df Deviance    AIC    LRT Pr(>Chi)
## <none>                  1378.3 1382.3
## Training_Intensity  1   1386.3 1388.3 7.9851 0.004716 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
exp(confint(g3))
```

```
## Waiting for profiling to be done...
```

```
##                        2.5 %    97.5 %
## (Intercept)        0.5733645 0.9412048
## Training_Intensity 1.2113880 2.9019183
```

Likelihood-Ratio Test

H0: B = 0

H1: B does not equal 0

LRT = 7.9851 df=1

p-value = 0.004716 < 0.05

We reject H0. There is significant evidence that beta does not equal 0.

The 95% Likelihood-Ratio CI is (1.2113880, 2.9019183). We are 95% confident that the the odds ratio corresponding to a 1-unit increase in training intensity has at least 21% increase and at most a 290% increase in odds of likelihood of injury.

4. Use the new model to make predictions

```
predict(g3, data.frame(Training_Intensity=mean(injury$Training_Intensity)), type="response")
```

```
##         1
## 0.5000048
```

```
#min,max,quartile

predict(g3, data.frame(Training_Intensity=min(injury$Training_Intensity)), type="response")
```

```
##         1
## 0.4237072
```

```
predict(g3, data.frame(Training_Intensity=max(injury$Training_Intensity)), type="response")
```

```
##         1
## 0.5788578
```

```
predict(g3, data.frame(Training_Intensity=quantile(injury$Training_Intensity)), type="response")
```

```
##        0%       25%       50%       75%      100%
## 0.4237072 0.4609701 0.4989659 0.5375385 0.5788578
```

```
pred.prob <- fitted(g3) # ML fitted value estimate of P(Y=1)
pred.prob
```

```
##         1         2         3         4         5         6         7         8 
## 0.4948927 0.4587085 0.5193459 0.4628119 0.5136553 0.4794276 0.5520233 0.5515804 
##         9        10        11        12        13        14        15        16 
## 0.4979373 0.4781147 0.4799520 0.5492555 0.4745733 0.4559908 0.4367198 0.4962666 
##        17        18        19        20        21        22        23        24 
## 0.4986869 0.5547133 0.4578860 0.4828376 0.5496733 0.5520344 0.4834044 0.4950801 
##        25        26        27        28        29        30        31        32 
## 0.4706824 0.5688605 0.4419647 0.5690442 0.5302763 0.5064374 0.4287109 0.4976623 
##        33        34        35        36        37        38        39        40 
## 0.4870285 0.4485592 0.5259855 0.5747864 0.5112358 0.5350978 0.4342166 0.4885826 
##        41        42        43        44        45        46        47        48 
## 0.4416113 0.5191621 0.5697719 0.5268247 0.4355038 0.4787167 0.5094297 0.4863172 
##        49        50        51        52        53        54        55        56 
## 0.5535878 0.5506457 0.5190110 0.4816580 0.4631983 0.4399907 0.4787529 0.4878783 
##        57        58        59        60        61        62        63        64 
## 0.5290197 0.5262591 0.4344720 0.4850721 0.4517504 0.4478758 0.5526048 0.4301331 
##        65        66        67        68        69        70        71        72 
## 0.4888201 0.4496429 0.5606565 0.5104290 0.4594651 0.5018687 0.5377071 0.5727877 
##        73        74        75        76        77        78        79        80 
## 0.4577626 0.5617146 0.5691295 0.5663774 0.5227018 0.5218266 0.4863176 0.5409652 
##        81        82        83        84        85        86        87        88 
## 0.5063938 0.5292721 0.4902735 0.5377087 0.5521659 0.4462673 0.5534771 0.5078049 
##        89        90        91        92        93        94        95        96 
## 0.5552952 0.4907192 0.4825321 0.5661679 0.4625486 0.5552250 0.4989830 0.5036979 
##        97        98        99       100       101       102       103       104 
## 0.4714922 0.5129363 0.4741092 0.4297265 0.4657973 0.4246795 0.5758901 0.5739580 
##       105       106       107       108       109       110       111       112 
## 0.4850558 0.5372215 0.4773303 0.5282806 0.5492114 0.5710433 0.4858479 0.5457730 
##       113       114       115       116       117       118       119       120 
## 0.4648502 0.5777236 0.4276531 0.5177112 0.5264695 0.5309290 0.4422344 0.5697792 
##       121       122       123       124       125       126       127       128 
## 0.4516579 0.5207206 0.4581098 0.4712966 0.5087707 0.4885084 0.4484051 0.4500670 
##       129       130       131       132       133       134       135       136 
## 0.4886574 0.5417258 0.5635328 0.4366333 0.4847345 0.4391211 0.4262451 0.5268154 
##       137       138       139       140       141       142       143       144 
## 0.5175422 0.4488385 0.4598440 0.4273367 0.5537897 0.5753288 0.4445670 0.4594799 
##       145       146       147       148       149       150       151       152 
## 0.5589690 0.5679235 0.4888897 0.4314720 0.4297453 0.5130785 0.4847018 0.4280957 
##       153       154       155       156       157       158       159       160 
## 0.5145357 0.4254341 0.5464063 0.4711741 0.4298784 0.5153345 0.4854520 0.5751924 
##       161       162       163       164       165       166       167       168 
## 0.5084165 0.4663030 0.5342686 0.4657182 0.5645010 0.4818560 0.5093051 0.4314644 
##       169       170       171       172       173       174       175       176 
## 0.4899073 0.5533893 0.5492453 0.4583655 0.4586396 0.5510544 0.5685288 0.4383669 
##       177       178       179       180       181       182       183       184 
## 0.4936590 0.4760238 0.5593212 0.4365403 0.4563123 0.5409762 0.4315753 0.5003179 
##       185       186       187       188       189       190       191       192 
## 0.4924127 0.4755463 0.4849645 0.5061817 0.4485822 0.5127723 0.5492117 0.5421738 
##       193       194       195       196       197       198       199       200 
## 0.4474242 0.4467033 0.4652006 0.4797196 0.4871393 0.5296250 0.4324042 0.4290202 
##       201       202       203       204       205       206       207       208 
## 0.4845476 0.5323527 0.4535614 0.5236545 0.4638987 0.5616998 0.5631819 0.4697453 
##       209       210       211       212       213       214       215       216 
## 0.4592492 0.4875856 0.4608907 0.5284823 0.5524129 0.5285930 0.5521471 0.4853435 
##       217       218       219       220       221       222       223       224 
## 0.4477990 0.5387147 0.4796257 0.5283084 0.4655859 0.4361863 0.5780676 0.4477811 
##       225       226       227       228       229       230       231       232 
## 0.5774311 0.5757255 0.5474079 0.5264566 0.5136827 0.5586121 0.4685197 0.4964215 
##       233       234       235       236       237       238       239       240 
## 0.5201942 0.4875678 0.4901213 0.4749025 0.5115559 0.5562100 0.4548195 0.5691494 
##       241       242       243       244       245       246       247       248 
## 0.5310917 0.5519800 0.5102958 0.5451851 0.4261850 0.5512212 0.4298601 0.5622906 
##       249       250       251       252       253       254       255       256 
## 0.5779730 0.4692424 0.4561867 0.5429835 0.4628381 0.5585288 0.4395223 0.4430958 
##       257       258       259       260       261       262       263       264 
## 0.5760121 0.5287097 0.5556392 0.4739714 0.5291230 0.5162569 0.5176519 0.5300679 
##       265       266       267       268       269       270       271       272 
## 0.5132993 0.4903683 0.4664096 0.5434842 0.4586697 0.5316018 0.4597692 0.5211239 
##       273       274       275       276       277       278       279       280 
## 0.5401367 0.4574930 0.4329064 0.4438725 0.5180977 0.5560354 0.4306069 0.5381225 
##       281       282       283       284       285       286       287       288 
## 0.4766283 0.4981220 0.5682921 0.4751663 0.4960540 0.4257989 0.4362432 0.4637059 
##       289       290       291       292       293       294       295       296 
## 0.4279728 0.5220698 0.4899508 0.5090350 0.4506430 0.4695337 0.5271446 0.5738898 
##       297       298       299       300       301       302       303       304 
## 0.4314404 0.5623633 0.5135377 0.5115069 0.5015318 0.4343619 0.4375204 0.5173015 
##       305       306       307       308       309       310       311       312 
## 0.4765720 0.5665079 0.4868471 0.4457785 0.5350924 0.4691309 0.5054810 0.5324572 
##       313       314       315       316       317       318       319       320 
## 0.5638875 0.5471552 0.5291038 0.5296150 0.5709156 0.4695168 0.4238728 0.4657904 
##       321       322       323       324       325       326       327       328 
## 0.4573650 0.5268336 0.5224354 0.5161347 0.4261916 0.5372849 0.4738464 0.5273371 
##       329       330       331       332       333       334       335       336 
## 0.5103566 0.4768492 0.4444302 0.4382214 0.5532583 0.5666818 0.5249501 0.4396137 
##       337       338       339       340       341       342       343       344 
## 0.4861323 0.5373101 0.5452324 0.4418989 0.4237072 0.5346896 0.4790187 0.4630649 
##       345       346       347       348       349       350       351       352 
## 0.4256786 0.5078054 0.5562992 0.5727131 0.5117930 0.5037552 0.4367766 0.5091490 
##       353       354       355       356       357       358       359       360 
## 0.4826791 0.5182759 0.4840489 0.4608544 0.4383058 0.4725002 0.4385651 0.4509923 
##       361       362       363       364       365       366       367       368 
## 0.5772651 0.4927256 0.5064844 0.5597741 0.5785245 0.5144515 0.5503047 0.4743797 
##       369       370       371       372       373       374       375       376 
## 0.4710665 0.4862382 0.5285016 0.5300147 0.4725504 0.4442187 0.5222296 0.4434626 
##       377       378       379       380       381       382       383       384 
## 0.5137901 0.5317686 0.5328910 0.5411325 0.5596568 0.5014817 0.5375143 0.5201602 
##       385       386       387       388       389       390       391       392 
## 0.4525055 0.4276409 0.4677549 0.4925579 0.5198367 0.5561148 0.4539983 0.4430330 
##       393       394       395       396       397       398       399       400 
## 0.5736103 0.4403260 0.4981161 0.5148978 0.5079738 0.4369271 0.4324575 0.4399303 
##       401       402       403       404       405       406       407       408 
## 0.5149015 0.5084394 0.4598327 0.5230710 0.5514829 0.4300814 0.5011950 0.5311265 
##       409       410       411       412       413       414       415       416 
## 0.4627267 0.4713832 0.5192939 0.5635283 0.5499090 0.5145166 0.5375100 0.4803062 
##       417       418       419       420       421       422       423       424 
## 0.5235702 0.4961864 0.4530176 0.5330668 0.5103388 0.4793357 0.5655946 0.4269111 
##       425       426       427       428       429       430       431       432 
## 0.4726338 0.4324351 0.5431646 0.5331245 0.4750695 0.5289947 0.4851783 0.5415656 
##       433       434       435       436       437       438       439       440 
## 0.4943056 0.4877385 0.5691950 0.4625033 0.4422436 0.5147379 0.5745060 0.4823440 
##       441       442       443       444       445       446       447       448 
## 0.4332287 0.4763134 0.5024180 0.4486684 0.5262501 0.5788578 0.4939225 0.4787264 
##       449       450       451       452       453       454       455       456 
## 0.4859250 0.4414238 0.5612655 0.4881374 0.4838215 0.5264984 0.4924012 0.5247070 
##       457       458       459       460       461       462       463       464 
## 0.4331378 0.5504986 0.5700827 0.5247901 0.5722425 0.4470417 0.4979476 0.5174416 
##       465       466       467       468       469       470       471       472 
## 0.5142983 0.4860437 0.4761224 0.4432302 0.4845213 0.4487829 0.5381559 0.4559553
```

```
##       473       474       475       476       477       478       479       480 
## 0.4326943 0.5096965 0.5164117 0.5595105 0.5135534 0.4772794 0.5488359 0.5078021 
##       481       482       483       484       485       486       487       488 
## 0.4358857 0.5118095 0.5650829 0.4852371 0.4695952 0.4458076 0.4470437 0.4909495 
##       489       490       491       492       493       494       495       496 
## 0.5164821 0.4360916 0.5699389 0.5413912 0.5150638 0.5527810 0.4360255 0.4978571 
##       497       498       499       500       501       502       503       504 
## 0.5218692 0.5528142 0.5457997 0.4665886 0.5701405 0.4428468 0.5600082 0.5747541 
##       505       506       507       508       509       510       511       512 
## 0.4510321 0.5362384 0.4298826 0.4866642 0.5040408 0.5141273 0.5693806 0.5284615 
##       513       514       515       516       517       518       519       520 
## 0.4985530 0.5498703 0.5716233 0.4270848 0.5765281 0.4369425 0.5506016 0.4670983 
##       521       522       523       524       525       526       527       528 
## 0.4440592 0.4920598 0.5240380 0.4828314 0.4320787 0.5170743 0.5642173 0.4831859 
##       529       530       531       532       533       534       535       536 
## 0.4571225 0.4924864 0.4311245 0.5515480 0.5524557 0.5152794 0.4784720 0.5483346 
##       537       538       539       540       541       542       543       544 
## 0.5100602 0.5523924 0.5220829 0.5459452 0.5169336 0.4880301 0.5727613 0.5079330 
##       545       546       547       548       549       550       551       552 
## 0.5179556 0.4577691 0.5211741 0.5127115 0.4522256 0.4328516 0.5178412 0.5427234 
##       553       554       555       556       557       558       559       560 
## 0.5051567 0.4587476 0.5275662 0.4360130 0.4923699 0.4488924 0.4521667 0.4549386 
##       561       562       563       564       565       566       567       568 
## 0.4837327 0.4315532 0.4854690 0.5033235 0.4988691 0.4831183 0.5544404 0.4460954 
##       569       570       571       572       573       574       575       576 
## 0.5023812 0.4332294 0.4346334 0.5130653 0.5118358 0.5604562 0.5105596 0.5730444 
##       577       578       579       580       581       582       583       584 
## 0.4311755 0.4387755 0.4304637 0.4522937 0.5079575 0.5242907 0.4306973 0.5709227 
##       585       586       587       588       589       590       591       592 
## 0.5549080 0.4696399 0.4355631 0.4499041 0.4432489 0.4426824 0.5043980 0.4617190 
##       593       594       595       596       597       598       599       600 
## 0.4792602 0.5775970 0.5303462 0.5713850 0.4456662 0.4830174 0.5100672 0.4354969 
##       601       602       603       604       605       606       607       608 
## 0.4243438 0.5281777 0.5237038 0.4875049 0.4998646 0.4882341 0.4273823 0.4756981 
##       609       610       611       612       613       614       615       616 
## 0.4510211 0.4388399 0.5725732 0.5170584 0.5372660 0.4725698 0.4847413 0.4603286 
##       617       618       619       620       621       622       623       624 
## 0.4386146 0.4513238 0.5479821 0.5292889 0.5088444 0.4975829 0.5673033 0.4348767 
##       625       626       627       628       629       630       631       632 
## 0.4671398 0.4779768 0.5456374 0.5780574 0.4609123 0.5598805 0.5531195 0.4584182 
##       633       634       635       636       637       638       639       640 
## 0.4857101 0.4873300 0.5758696 0.4516126 0.5482374 0.4755137 0.5376111 0.4889499 
##       641       642       643       644       645       646       647       648 
## 0.5137693 0.5534278 0.5490151 0.5586619 0.4328562 0.5315544 0.4452682 0.4882929 
##       649       650       651       652       653       654       655       656 
## 0.5091045 0.4862438 0.5045626 0.5786708 0.4445295 0.5289624 0.4851883 0.4441877 
##       657       658       659       660       661       662       663       664 
## 0.4481742 0.5714299 0.5606984 0.5649324 0.5779072 0.4558751 0.4788376 0.5279878 
##       665       666       667       668       669       670       671       672 
## 0.4989488 0.4886958 0.4792088 0.5161847 0.5133360 0.4485963 0.4970578 0.5098698 
##       673       674       675       676       677       678       679       680 
## 0.5125046 0.4561538 0.5393879 0.4275656 0.4787426 0.5452933 0.5115028 0.4641068 
##       681       682       683       684       685       686       687       688 
## 0.5320076 0.5120509 0.5477847 0.5382587 0.5187413 0.4996347 0.4442144 0.4641334 
##       689       690       691       692       693       694       695       696 
## 0.4888070 0.5170489 0.5036371 0.4683238 0.4246926 0.5008986 0.4679243 0.5381938 
##       697       698       699       700       701       702       703       704 
## 0.4273733 0.5147825 0.5702318 0.4506061 0.4970436 0.4377588 0.5211789 0.5094090 
##       705       706       707       708       709       710       711       712 
## 0.4869685 0.5048393 0.5633184 0.4540058 0.4271613 0.5579260 0.5135747 0.5624056 
##       713       714       715       716       717       718       719       720 
## 0.5166803 0.5499458 0.4904640 0.5405700 0.5658569 0.5127923 0.4515961 0.4652978 
##       721       722       723       724       725       726       727       728 
## 0.4543712 0.4618764 0.4711714 0.5101302 0.5152781 0.4899667 0.5195730 0.4362396 
##       729       730       731       732       733       734       735       736 
## 0.4371981 0.4502056 0.5043221 0.4567586 0.4675587 0.4858637 0.5502104 0.4258265 
##       737       738       739       740       741       742       743       744 
## 0.5248861 0.5279342 0.5481594 0.5688910 0.4267889 0.4474062 0.5617130 0.4949852 
##       745       746       747       748       749       750       751       752 
## 0.5116121 0.5270200 0.5295265 0.5687352 0.5787355 0.5359407 0.4681733 0.5607217 
##       753       754       755       756       757       758       759       760 
## 0.4312565 0.4594185 0.5634435 0.4633316 0.4577946 0.4449798 0.5575311 0.5014973 
##       761       762       763       764       765       766       767       768 
## 0.5288744 0.4606624 0.5418437 0.5421496 0.4721375 0.4874882 0.4560889 0.5756154 
##       769       770       771       772       773       774       775       776 
## 0.5242955 0.5691664 0.4706980 0.5455542 0.5430606 0.5314928 0.5739414 0.4848052 
##       777       778       779       780       781       782       783       784 
## 0.4438328 0.5283769 0.5373678 0.5132525 0.4556421 0.5021867 0.4720737 0.4901182 
##       785       786       787       788       789       790       791       792 
## 0.5242354 0.5398464 0.4595966 0.4866489 0.4653546 0.4543692 0.5457869 0.5652731 
##       793       794       795       796       797       798       799       800 
## 0.4811530 0.5407909 0.5633833 0.5555210 0.4519111 0.5725074 0.4821620 0.4763032 
##       801       802       803       804       805       806       807       808 
## 0.4332895 0.4993206 0.4404758 0.4621387 0.4728166 0.5497386 0.5600630 0.4705544 
##       809       810       811       812       813       814       815       816 
## 0.5574671 0.5572401 0.5428726 0.4463074 0.5312846 0.5707772 0.5300692 0.4751109 
##       817       818       819       820       821       822       823       824 
## 0.5015790 0.5243146 0.5031453 0.4421672 0.4720541 0.5487403 0.5578879 0.4448520 
##       825       826       827       828       829       830       831       832 
## 0.5719080 0.4746739 0.5269085 0.5408760 0.5502228 0.5708887 0.4721878 0.5574375 
##       833       834       835       836       837       838       839       840 
## 0.4439829 0.5335823 0.5393421 0.5293166 0.4627272 0.5596852 0.4497445 0.4319355 
##       841       842       843       844       845       846       847       848 
## 0.5355208 0.4976434 0.5559815 0.4832143 0.4496545 0.5540214 0.5091548 0.4529026 
##       849       850       851       852       853       854       855       856 
## 0.5359760 0.5032304 0.5178340 0.4953582 0.5528229 0.5529552 0.5342608 0.4399176 
##       857       858       859       860       861       862       863       864 
## 0.5543950 0.5282681 0.5323310 0.4800183 0.5606487 0.5367538 0.4410287 0.4870332 
##       865       866       867       868       869       870       871       872 
## 0.5617452 0.4309451 0.5732938 0.5295415 0.5220699 0.4655727 0.4621780 0.5703215 
##       873       874       875       876       877       878       879       880 
## 0.4698992 0.5445613 0.4847373 0.4451546 0.4253440 0.5084257 0.5754955 0.4809815 
##       881       882       883       884       885       886       887       888 
## 0.4791120 0.4770621 0.5047087 0.4921867 0.5607468 0.5023536 0.5027043 0.4450180 
##       889       890       891       892       893       894       895       896 
## 0.5765615 0.4623621 0.4349737 0.5270386 0.5164331 0.4881875 0.4892917 0.5460475 
##       897       898       899       900       901       902       903       904 
## 0.4539154 0.5515117 0.5247873 0.4887867 0.4949084 0.4634027 0.5409210 0.5035065 
##       905       906       907       908       909       910       911       912 
## 0.4734281 0.5165401 0.5785498 0.4913399 0.5670350 0.4772068 0.4321044 0.4552415 
##       913       914       915       916       917       918       919       920 
## 0.4887437 0.4576364 0.4237372 0.5661588 0.5551734 0.5393911 0.4495825 0.4422307 
##       921       922       923       924       925       926       927       928 
## 0.4340863 0.5365445 0.5682842 0.4905992 0.4420415 0.5034309 0.4778037 0.4818721 
##       929       930       931       932       933       934       935       936 
## 0.5204980 0.5357162 0.4407216 0.4291612 0.5783351 0.4596750 0.4319656 0.4530082 
##       937       938       939       940       941       942       943       944 
## 0.4296786 0.4845320 0.5487412 0.4766167 0.4927381 0.5289828 0.5028825 0.5578921 
```

```
##       945       946       947       948       949       950       951       952
## 0.5586019 0.4252207 0.5449565 0.4953324 0.5774953 0.4988594 0.5771978 0.5413033
##       953       954       955       956       957       958       959       960
## 0.4621488 0.5075069 0.5779994 0.5572865 0.4386766 0.5337900 0.4772949 0.4247719
##       961       962       963       964       965       966       967       968
## 0.4863600 0.4599906 0.5512970 0.4760045 0.5526391 0.4237940 0.4713907 0.4609894
##       969       970       971       972       973       974       975       976
## 0.4598057 0.4341759 0.4525807 0.5101954 0.4688198 0.4885399 0.4821918 0.5601274
##       977       978       979       980       981       982       983       984
## 0.5646638 0.5003840 0.4748824 0.4401338 0.5720640 0.4329501 0.4698020 0.4698670
##       985       986       987       988       989       990       991       992
## 0.4733572 0.4490351 0.5457518 0.4425608 0.5226224 0.4427694 0.4345378 0.5653004
##       993       994       995       996       997       998       999      1000
## 0.4530476 0.5516663 0.5084842 0.4356662 0.5723780 0.4618926 0.5576136 0.4497370
```

5. Use different pi_0 as a cut-off point and create a confusion table.

```
prop <- sum(injury$Likelihood_of_Injury)/nrow(injury) # sample proportion of 1's for y variable
prop
```

```
## [1] 0.5
```

```
predicted <- as.numeric(fitted(g3) > prop) # predict y=1 when est.> 0.5
 xtabs(~ injury$Likelihood_of_Injury + predicted)
```

```
##                              predicted
## injury$Likelihood_of_Injury   0   1
##                             0 273 227
##                             1 231 269
```

```
acc1 = (273+269)/1000
acc1
```

```
## [1] 0.542
```

```
predicted2<- as.numeric(fitted(g3) > 0.55)
xtabs(~ injury$Likelihood_of_Injury + predicted2)
```

```
##                              predicted2
## injury$Likelihood_of_Injury   0   1
##                             0 418  82
##                             1 399 101
```

```
acc2 = (418+101)/1000
acc2
```

```
## [1] 0.519
```

```
predicted3<- as.numeric(fitted(g3) > 0.45)
xtabs(~ injury$Likelihood_of_Injury + predicted3)
```

```
##                              predicted3
## injury$Likelihood_of_Injury   0   1
##                             0 104 396
##                             1  75 425
```

```
acc3 = (104+425)/1000
acc3
```

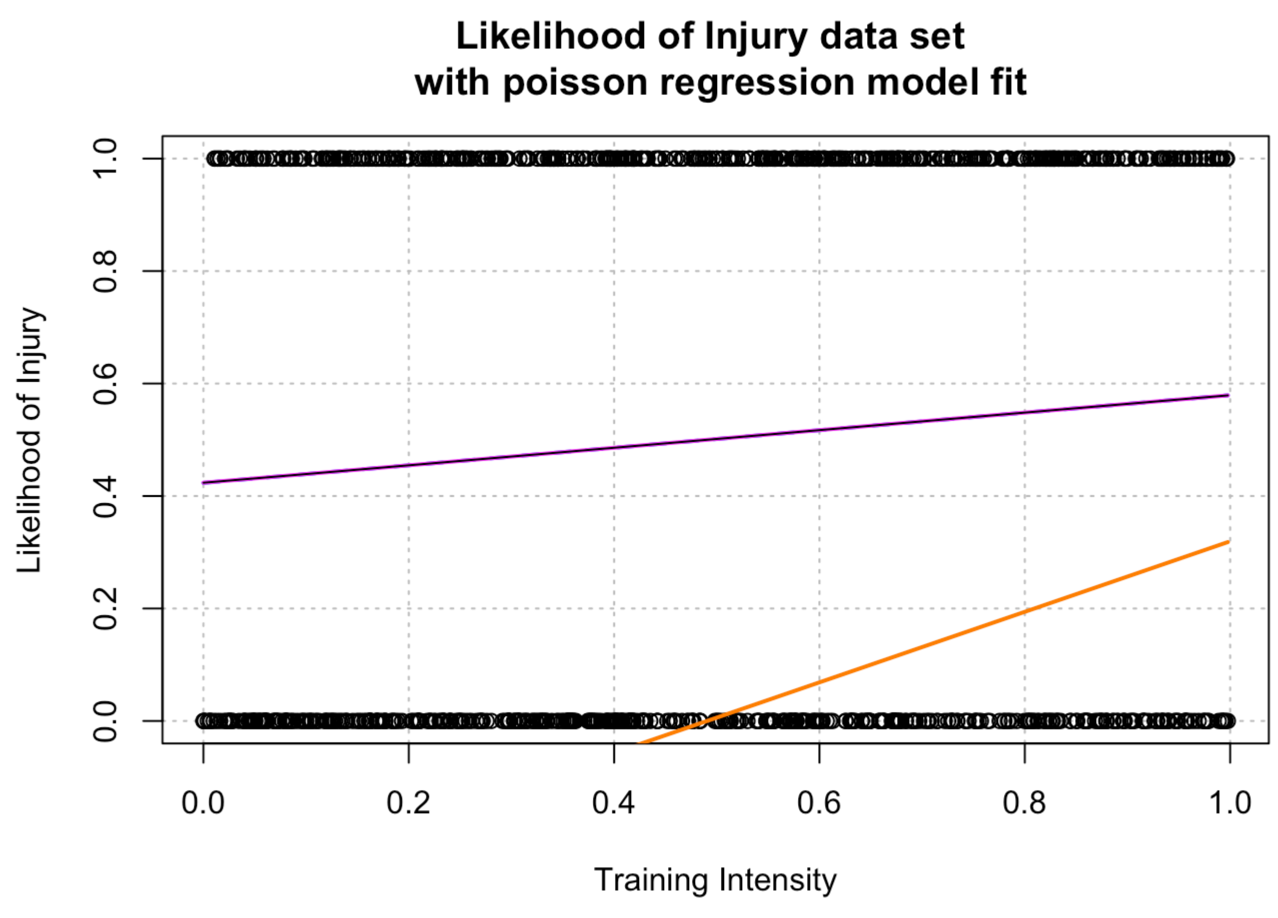
```
## [1] 0.529
```

The cut off of 0.5 has the highest accuracy of the three cut off points.

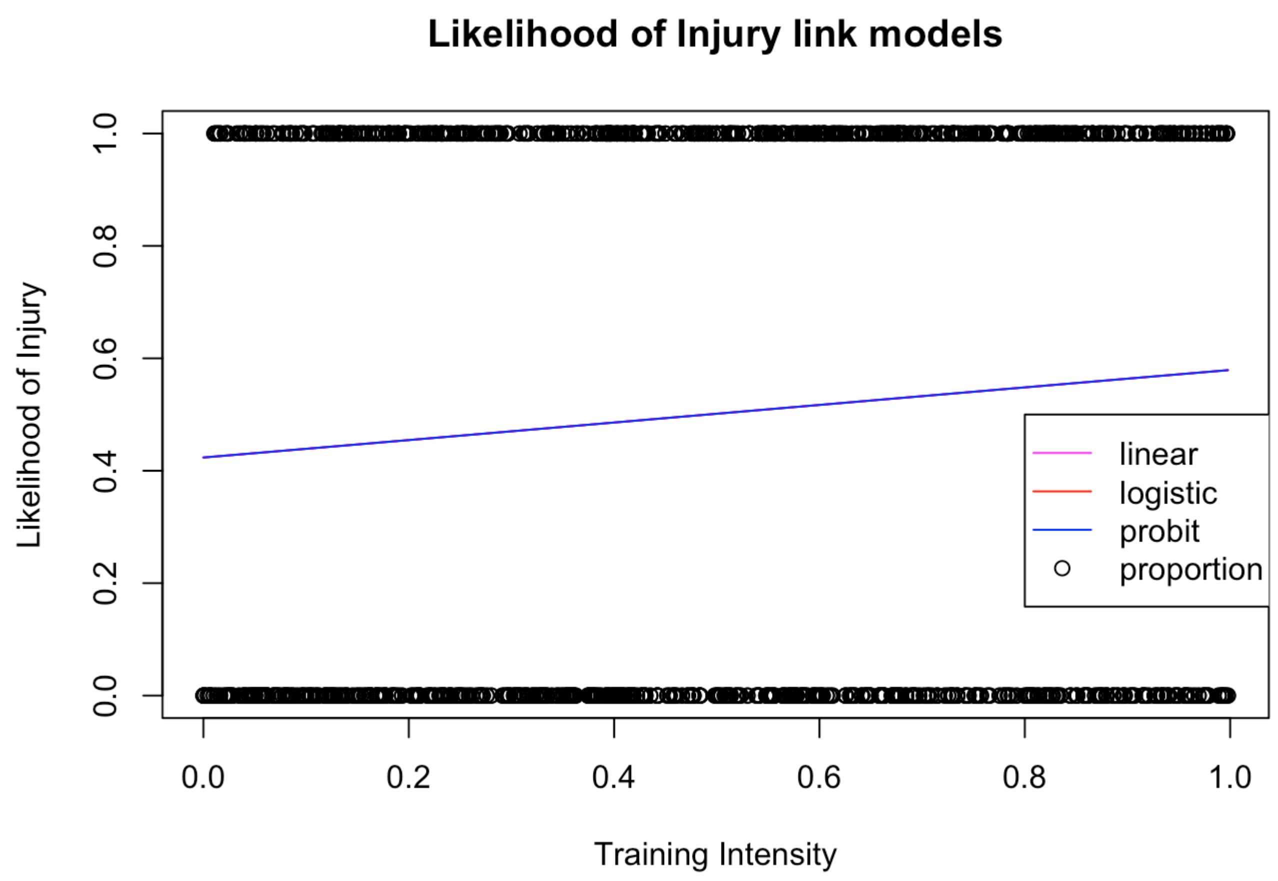
6. Perform visualization of data and models.

```
g8 = glm(formula = Likelihood_of_Injury ~ Training_Intensity, family = binomial(link = "identity"),
    data = injury)
g9 = glm(formula = Likelihood_of_Injury ~ Training_Intensity, family = binomial(link = "probit"),
    data = injury)
```

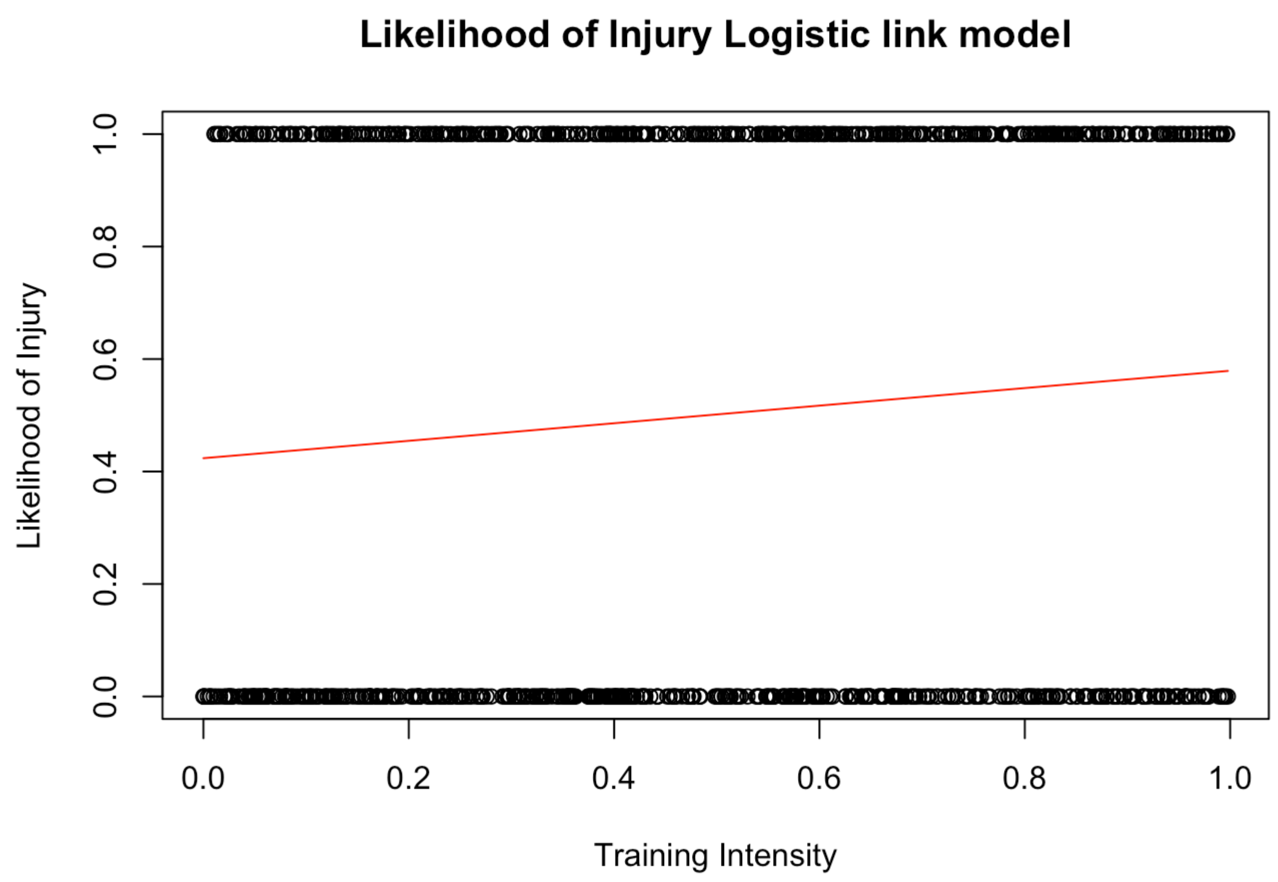
```
plot(x = injury$Training_Intensity, y = injury$Likelihood_of_Injury, xlab = "Training Intensity", ylab = "Likelih
ood of Injury",
     main = "Likelihood of Injury data set \n with poisson regression model fit", panel.first =
       grid(col = "gray", lty = "dotted"))
curve(expr = coef(g3)[1]+coef(g3)[2]*x, col = "darkorange1", add = TRUE, lty = 1, lwd=2)
curve(expr = coef(g8)[1]+coef(g8)[2]*x, col = "dodgerblue1", add = TRUE, lty = 2, lwd=2)
curve(predict(g9, data.frame(Training_Intensity=x),type="response"), col = "magenta", add = TRUE, lty = 1, lwd=2)
curve(predict(g9, data.frame(Training_Intensity=x), type="resp"), add=TRUE)
legend(22, 15, c("data", "linear regression", "ident", "poisson"), lty=c(-1,1,2,1), pch=c(1, -1,-1, -1),col=c("bl
ack", "darkorange1","dodgerblue1" , "magenta"))
```



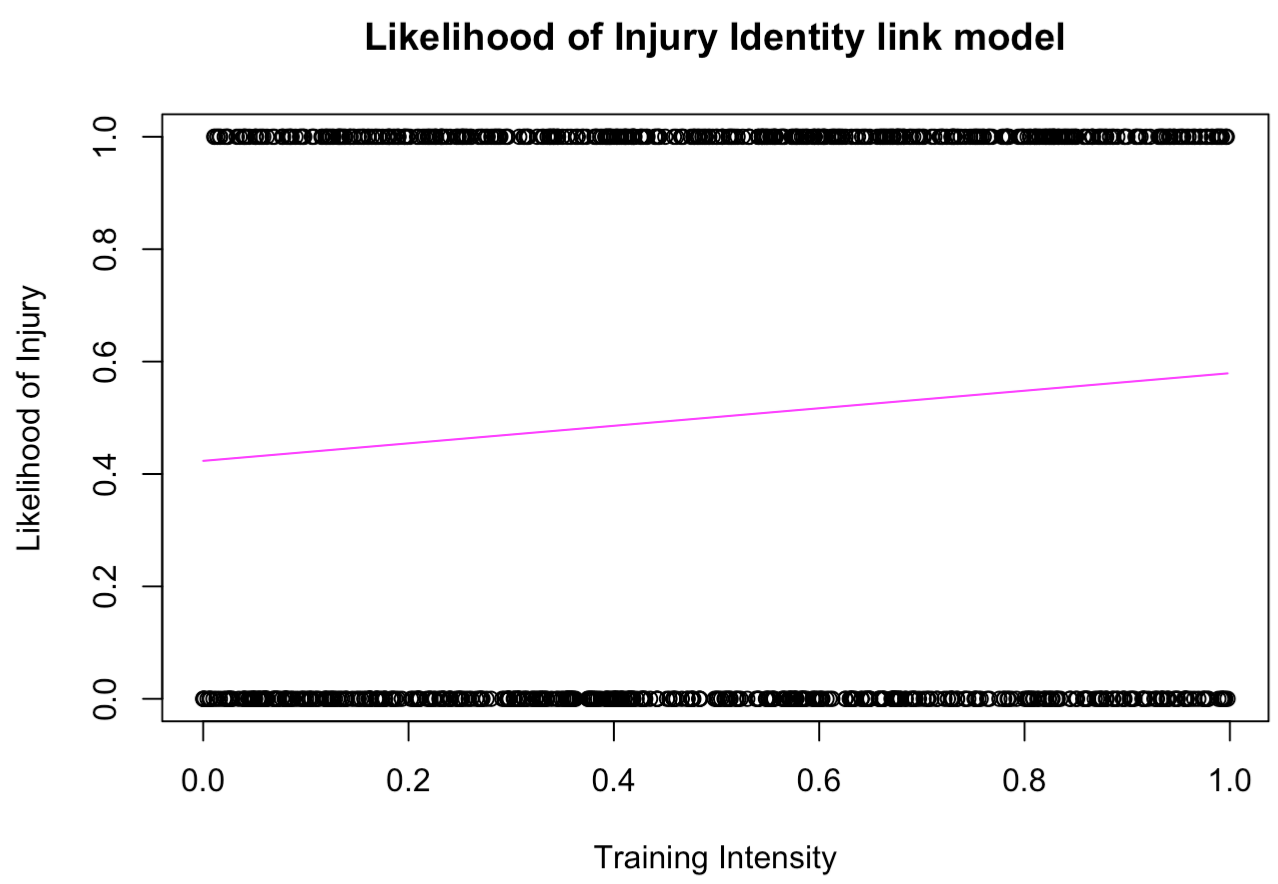
```
plot(x = injury$Training_Intensity, y = injury$Likelihood_of_Injury, xlab = "Training Intensity", ylab = "Likelih
ood of Injury", main = "Likelihood of Injury link models")
#lines(Training_Intensity=x,y = injury$Likelihood_of_Injury, g8, lty=1)
#curve(expr = plogis(g3$coefficients[1]+g3$coefficients[2]*x), col = "red", add = TRUE)
curve(predict(g3, data.frame(Training_Intensity=x),type="response"),  col="magenta",add = TRUE)
curve(predict(g8, data.frame(Training_Intensity=x),type="response"),  col="red",add = TRUE)
curve(predict(g9, data.frame(Training_Intensity=x),type="response"),  col="blue",add = TRUE)
points(x = injury$Training_Intensity, y = injury$Likelihood_of_Injury, pch=1)
legend(.8,.5,  c("linear", "logistic", "probit", "proportion"), lty=c(1,1,1,-1), pch=c(-1, -1, -1, 1), col=c("mag
enta", "red", "blue", "black"))
```



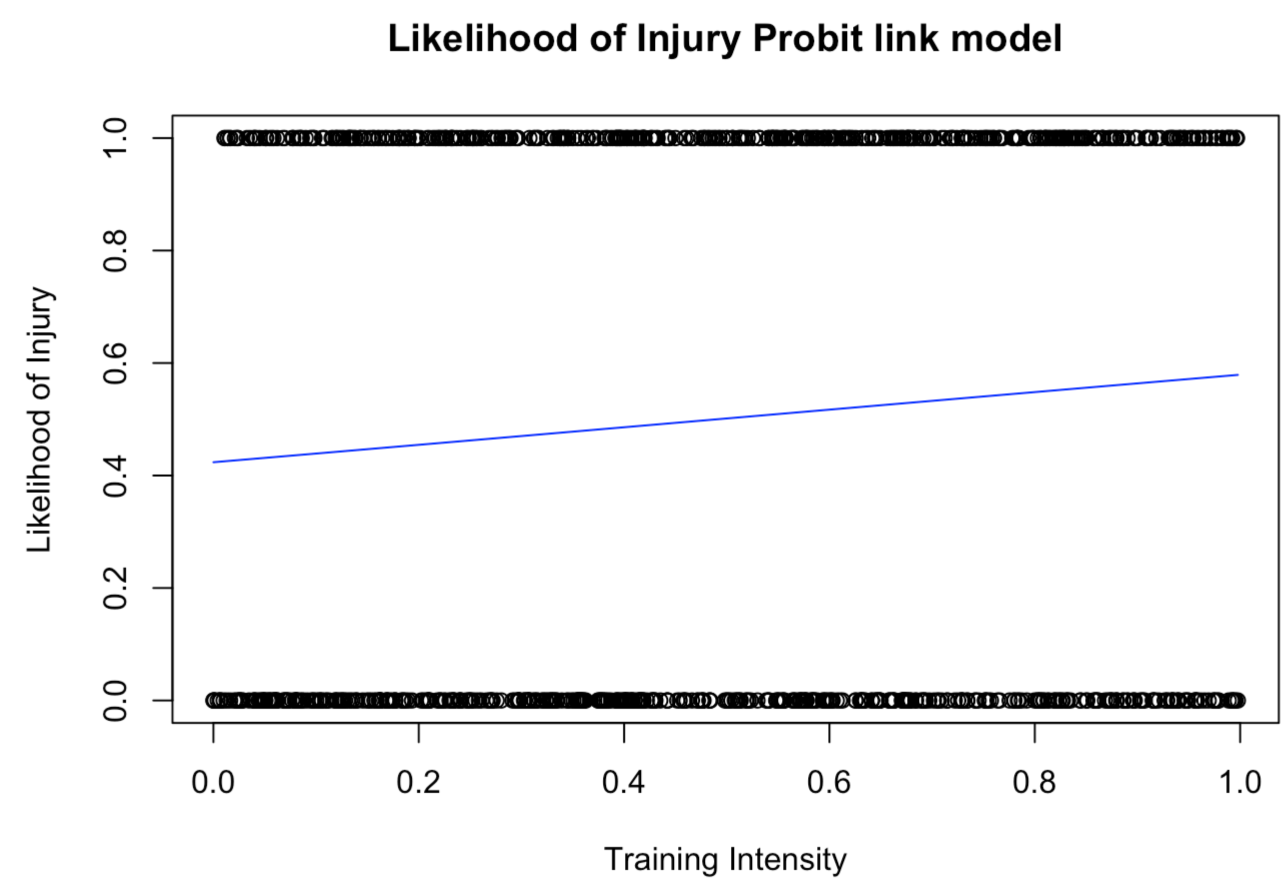

```
plot(x = injury$Training_Intensity, y = injury$Likelihood_of_Injury, xlab = "Training Intensity", ylab = "Likelih
ood of Injury", main = "Likelihood of Injury Logistic link model")
curve(predict(g3, data.frame(Training_Intensity=x),type="response"),  col="red",add = TRUE)
```




```
plot(x = injury$Training_Intensity, y = injury$Likelihood_of_Injury, xlab = "Training Intensity", ylab = "Likelih
ood of Injury", main = "Likelihood of Injury Identity link model")
curve(predict(g8, data.frame(Training_Intensity=x),type="response"),  col="magenta",add = TRUE)
```

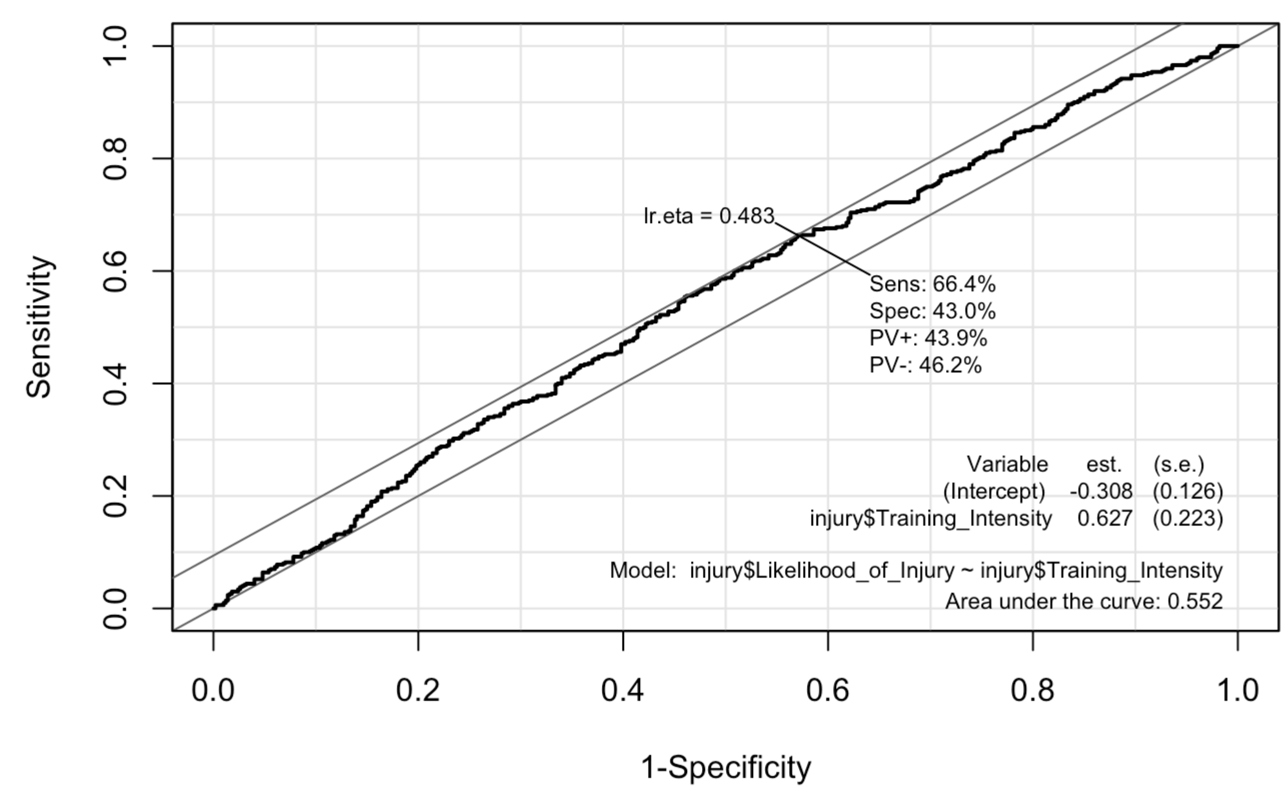



```
plot(x = injury$Training_Intensity, y = injury$Likelihood_of_Injury, xlab = "Training Intensity", ylab = "Likelih
ood of Injury", main = "Likelihood of Injury Probit link model")
curve(predict(g9, data.frame(Training_Intensity=x),type="response"),  col="blue",add = TRUE)
```



7. Plot the ROC curve, find AUC, and the best cutoff point for classification.

```
library(Epi)
ROC(form=injury$Likelihood_of_Injury ~ injury$Training_Intensity,plot="ROC")
```



AUC = .552 which is close to .50. The model is weak and barely better than a random guess.

The best cutoff point is 0.483

```
predicted4<- as.numeric(fitted(g3) >0.483)
xtabs(~ injury$Likelihood_of_Injury + predicted4)
```

```
##                              predicted4
## injury$Likelihood_of_Injury   0   1
##                             0 215 285
##                             1 169 331
```

```
acc4 = (215+331)/1000
acc4
```

```
## [1] 0.546
```

The accuracy with the cut off point of 0.483 is 0.546.

8. Perform LOOCV and k-fold cross-validation.

```
#LOOCV
pihat <- vector(length=1000)
 for (i in 1:1000) {
   pihat[i] <-
     predict(update(g3, subset=-i),
             newdata=injury[i,], type="response")
 }

 yy <- as.numeric( injury$Likelihood_of_Injury > 0)
 yhat <- as.numeric(pihat >prop)
 confusion <- table(yy, yhat)
 confusion
```

```
##    yhat
## yy    0   1
##   0 273 227
##   1 233 267
```

```
acc = (273+267)/1000
acc
```

```
## [1] 0.54
```

```
#k-fold
 library(lattice)
 library(DAAG)
```

```
##
## Attaching package: 'DAAG'
```

```
## The following object is masked from 'package:MASS':
##
##     hills
```

```
cv.binary(g3)
```

```
##
## Fold:  6 3 5 7 8 1 2 4 9 10
## Internal estimate of accuracy = 0.542
## Cross-validation estimate of accuracy = 0.544
```

The LOOCV accuracy is .54.

The K-fold Cross-validation accuracy is .538

LOOCV is more accurate.

9.Try the probit link and the identity links to model data.

```
injury.lin = glm(formula = Likelihood_of_Injury ~ Player_Age + Player_Weight + Player_Height + Previous_Injuries
+ Training_Intensity + Recovery_Time, family=binomial(link="identity"),
    data = injury)
g5 = summary(injury.lin)
g5
```

```
##
## Call:
## glm(formula = Likelihood_of_Injury ~ Player_Age + Player_Weight +
##     Player_Height + Previous_Injuries + Training_Intensity +
##     Recovery_Time, family = binomial(link = "identity"), data = injury)
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         0.2191067  0.3147270   0.696  0.48632
## Player_Age         -0.0002753  0.0024176  -0.114  0.90934
## Player_Weight      -0.0002867  0.0015980  -0.179  0.85760
## Player_Height       0.0012588  0.0015965   0.788  0.43043
## Previous_Injuries   0.0398073  0.0315265   1.263  0.20671
## Training_Intensity  0.1550602  0.0550110   2.819  0.00482 **
## Recovery_Time      -0.0037404  0.0092868  -0.403  0.68712
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1386.3  on 999  degrees of freedom
## Residual deviance: 1375.9  on 993  degrees of freedom
## AIC: 1389.9
##
## Number of Fisher Scoring iterations: 3
```

```
injury.probit = glm(formula = Likelihood_of_Injury ~ Player_Age + Player_Weight + Player_Height + Previous_Injuri
es + Training_Intensity + Recovery_Time, family=binomial(link="probit"),
    data = injury)
g6 = summary(injury.probit)
g6
```

```
##
## Call:
## glm(formula = Likelihood_of_Injury ~ Player_Age + Player_Weight +
##     Player_Height + Previous_Injuries + Training_Intensity +
##     Recovery_Time, family = binomial(link = "probit"), data = injury)
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.7276511  0.7959019  -0.914  0.36059
## Player_Age         -0.0006818  0.0061101  -0.112  0.91115
## Player_Weight      -0.0006725  0.0040401  -0.166  0.86779
## Player_Height       0.0032633  0.0040347   0.809  0.41863
## Previous_Injuries   0.1002299  0.0796946   1.258  0.20851
## Training_Intensity  0.3910398  0.1397728   2.798  0.00515 **
## Recovery_Time      -0.0095060  0.0234702  -0.405  0.68546
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1386.3  on 999  degrees of freedom
## Residual deviance: 1375.9  on 993  degrees of freedom
## AIC: 1389.9
##
## Number of Fisher Scoring iterations: 3
```

10. Which model works better for this data?

```
anova(injury.lin, injury.probit, test="LRT")
```

```
## Analysis of Deviance Table
##
## Model 1: Likelihood_of_Injury ~ Player_Age + Player_Weight + Player_Height +
##     Previous_Injuries + Training_Intensity + Recovery_Time
## Model 2: Likelihood_of_Injury ~ Player_Age + Player_Weight + Player_Height +
##     Previous_Injuries + Training_Intensity + Recovery_Time
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       993     1375.9
## 2       993     1375.9  0 0.012225
```

Deviance is 0.012225, which means neither model is better than the other one

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

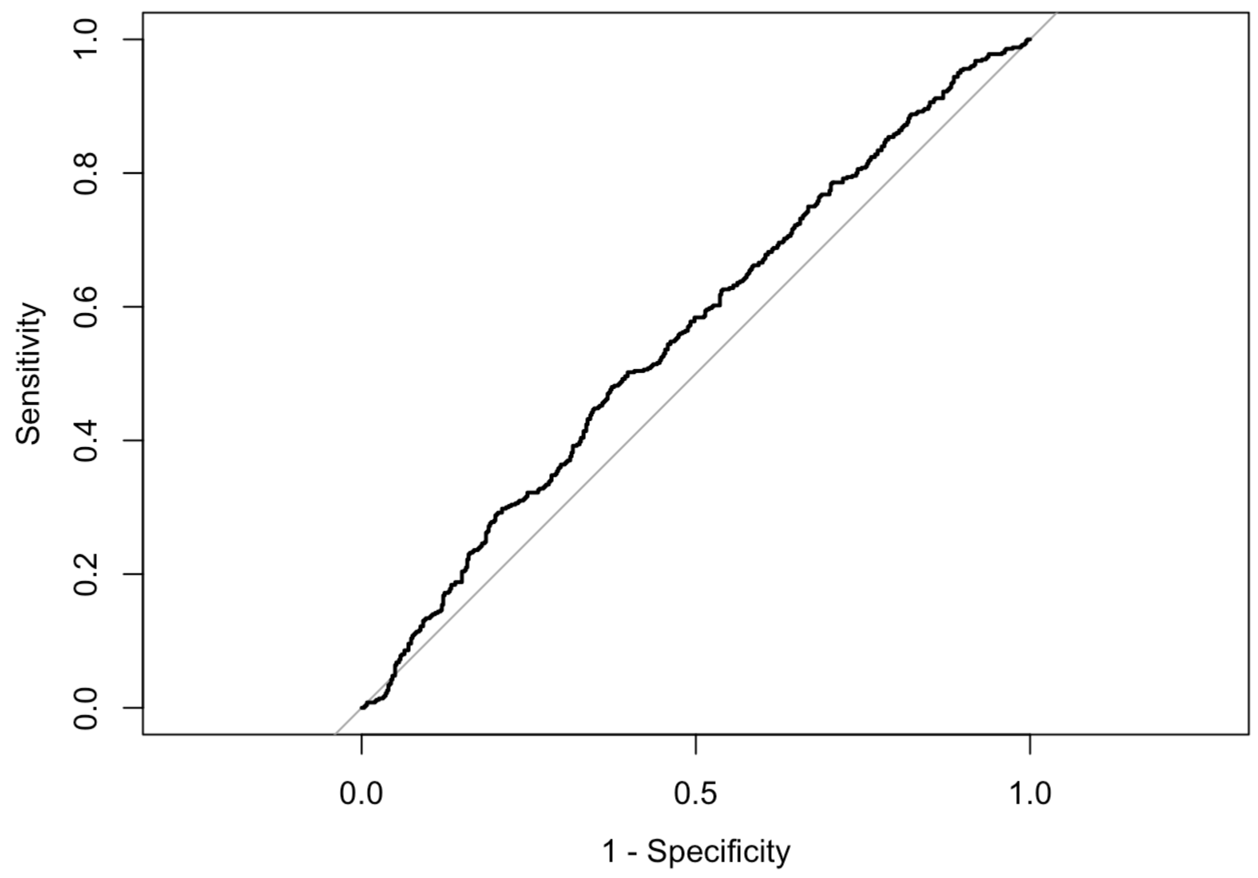
```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
rocplot0 <- roc(Likelihood_of_Injury ~ fitted(injury.lin), data=injury)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot.roc(rocplot0, legacy.axes=TRUE) # Specficity on x axis if legacy.axes=F
```



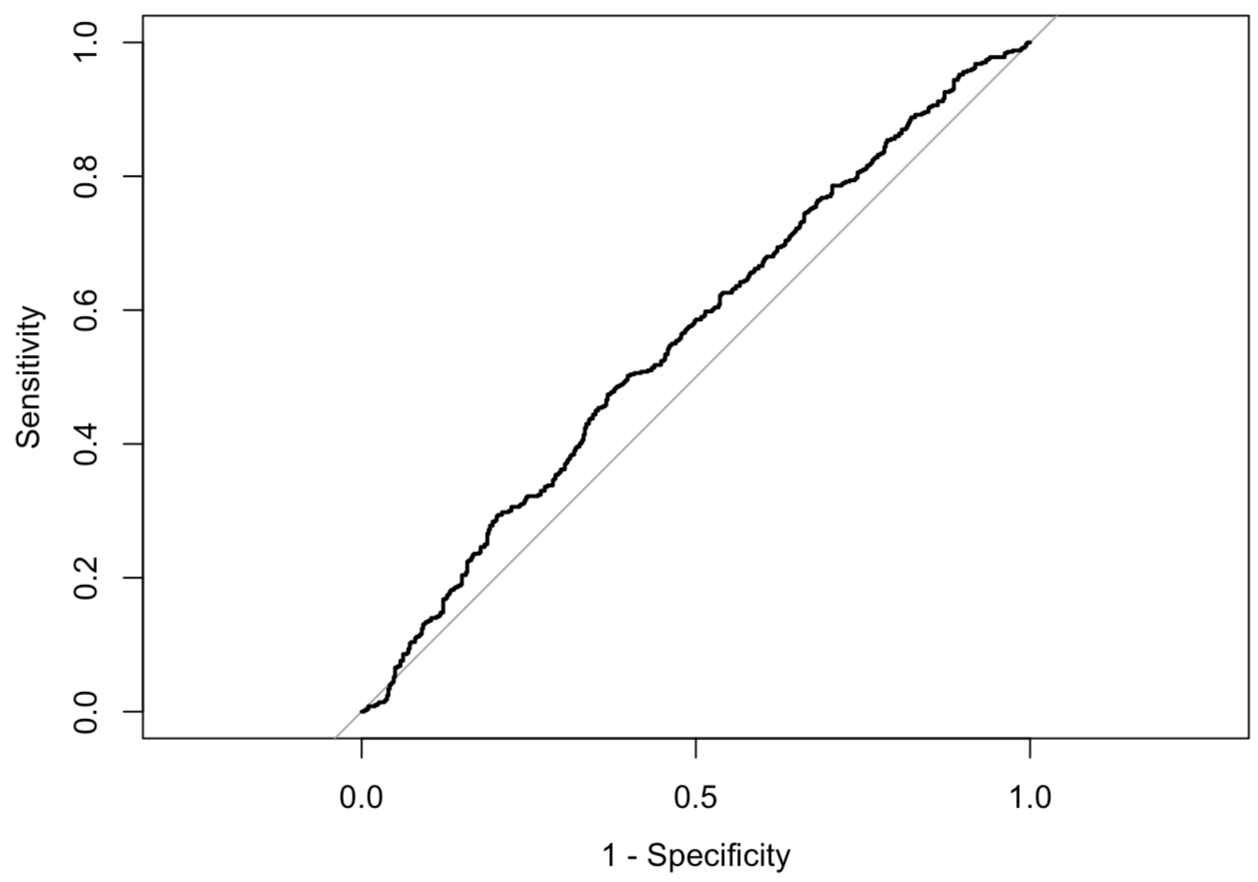

```
a1 = auc(rocplot0)
a1
```

```
## Area under the curve: 0.5592
```

```
rocplot1 <- roc(Likelihood_of_Injury ~ fitted(injury.probit), data=injury)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot.roc(rocplot1, legacy.axes=TRUE) # Specficity on x axis if legacy.axes=F
```



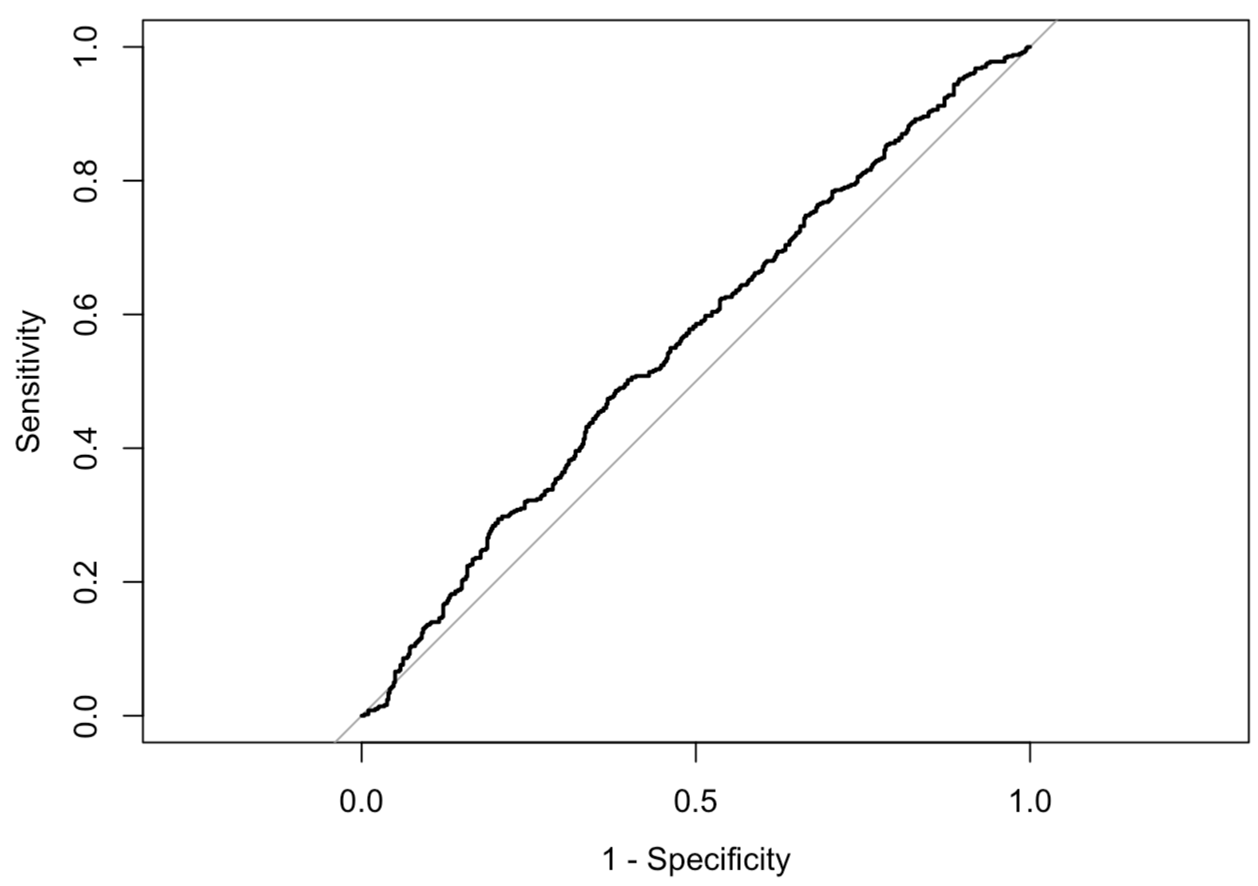

```
a2 = auc(rocplot1)
a2
```

```
## Area under the curve: 0.5594
```

```
rocplot2 <- roc(Likelihood_of_Injury ~ fitted(g1), data=injury)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot.roc(rocplot2, legacy.axes=TRUE) # Specficity on x axis if legacy.axes=F
```




```
a3 = auc(rocplot2)
a3
```

```
## Area under the curve: 0.5594
```