$$L_X(\theta) = \sum [y_i \log(p_i) + (1-y_i)\log(1-p_i)]$$

$$f(\beta_0, \beta_1, \ldots \beta_{30}) = \sum_{i=1}^{n} \left(y_i (\beta_0 + \beta_1 x_i + \ldots + \beta_{30} x_1) - \log(1 + e^{(\beta_0 + \ldots + \beta_{30} x_i)}) \right)$$

$$\nabla f(\beta_0, \ldots, \beta_{30}) = \begin{pmatrix} \sum y_i - p_i \\ \sum_{i=1}^{n} x_1 (y_i - p_i) \\ \vdots \\ \sum_{i=1}^{n} x_i (y_i - p_i) \end{pmatrix} = \sum x_i (y_i - p_i)$$

$$= X^T (y - p)$$

$$\nabla^2 f(\beta_0, \ldots, \beta_{30}) = -\sum_i \pi_i (1-\pi_i) x_i x_i^T$$

$$= -X^T \operatorname{diag}(\pi_i (1-\pi_i)) X = -X^T W X$$

where $\pi_i = \operatorname{sigm}(x_i, \theta)$

$$\beta_{new} = \beta_{old} - \frac{\nabla f}{\nabla^2 f}$$

X: 455×3) (fixed)

p: 455×1) (updated)

y: 455×1) (fixed)

Logistic Newton Raphson Full Model

Hun

3/17/2022

```
cancer_df <- read.csv("~/Downloads/breast-cancer.csv") %>% janitor::clean_names()
```

```
data <-
  cancer_df %>% dplyr::select(-id, -x) %>%
  mutate(diagnosis = ifelse(diagnosis == "M", 1, 0)) %>% distinct()

set.seed(7777)
split <- initial_split(data, prop = 0.8)

training_df <- split %>% training()

testing_df <- split %>% testing()
```

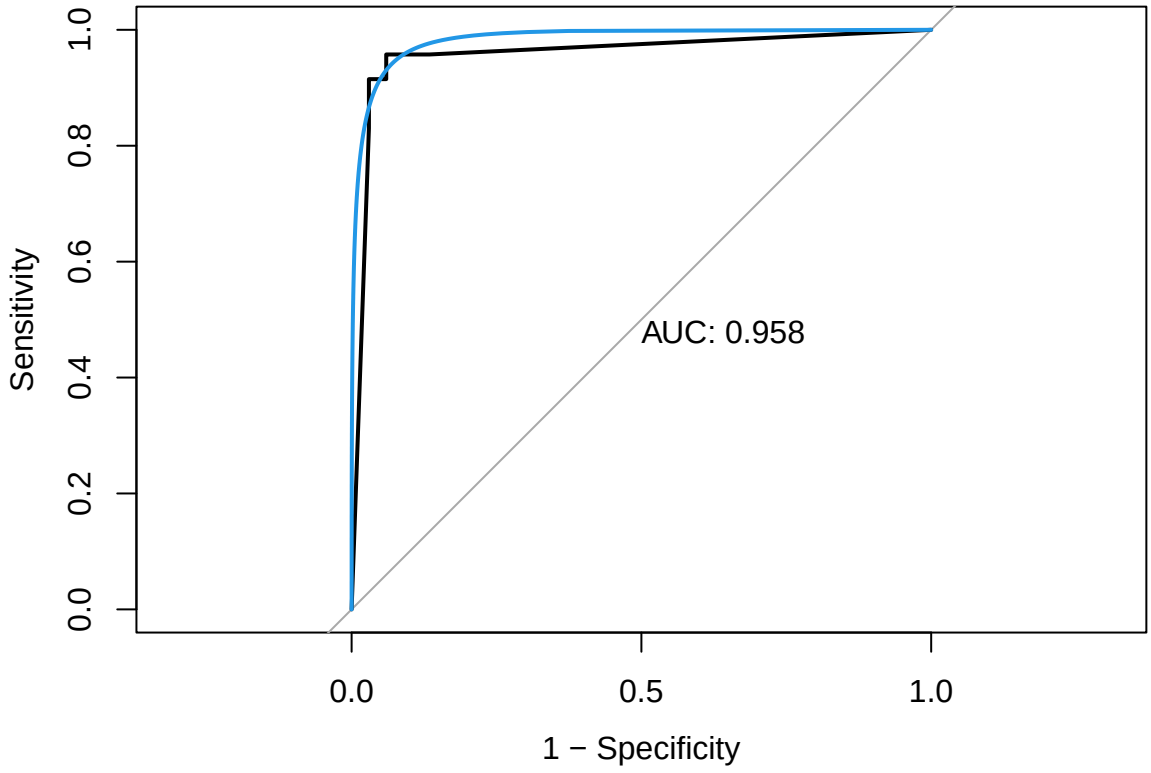
```
training_df_5p <- training_df %>% dplyr::select(1:5)
training_df_31p <- training_df

model_5p <- glm(diagnosis ~ ., data = training_df_5p, family = "binomial")
model_31p <- glm(diagnosis ~ ., data = training_df_31p, family = "binomial")
```

```
beta1 <- model_5p$coefficients %>% round(digits = 3) %>% broom::tidy()
```

```
beta2 <- model_31p$coefficients %>% round(digits = 3) %>% broom::tidy()
```

```
test_pred_prob <- predict(model_31p, testing_df, type = "response")
roc.glm <- roc(testing_df$diagnosis, test_pred_prob)
plot(roc.glm, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.glm), col = 4, add = TRUE)
```



Function for log likelihood, gradient, and Hessian

```
logisticstuff <- function(X, y, beta) {
  p <- exp(X %*% beta) / (1+ exp(X %*% beta)) %>% as.vector()
  for (i in 1:length(p)) {
    if (p[i] == 1) {
      p[i] <- 1-1e-8
    }
  }
  loglik <- t(y) %*% log(p) + t(1-y)  %*% log(1-p)
  grad <- t(X) %*% (y-p)
  W <- diag(c(p*(1-p)))
  Hess <- -t(X) %*% W %*% X
  return(list(loglik = loglik, grad = grad, Hess = Hess))
}
```

Newton Raphson with 5 parameters

```
X <- model.matrix(diagnosis~., training_df_5p)
y <- as.matrix(training_df$diagnosis)

NewtonRaphson <- function(X, y, logit_func, start, tol=1e-10, maxiter = 200) {
   i <- 0
   cur_beta <- start
   stuff <- logit_func(X, y, cur_beta)
   asc_dir_check <- -t(stuff$grad) %*% solve(stuff$Hess) %*% stuff$grad
   res <- c(i, stuff$loglik, asc_dir_check, cur_beta)
   prevloglik <- -Inf # To make sure it iterates

   while (i < maxiter && abs(stuff$loglik - prevloglik) > tol) {
     i <- i + 1
     prevloglik <- stuff$loglik
     prev_beta <- cur_beta
     cur_beta <- prev_beta - (solve(stuff$Hess) %*% stuff$grad) #update beta
     stuff <- logit_func(X, y, cur_beta) #update log likelihood, gradient, Hessia
     asc_dir_check <- -t(stuff$grad) %*% solve(stuff$Hess) %*% stuff$grad
     res <- rbind(res, c(i, stuff$loglik, asc_dir_check, cur_beta))
     colnames(res) <- c("Number of trial", "Log_likelihood", "asc_dir_check", paste0("Beta", 0:4))

   }
   return(res)
}

coef <- rep(0,ncol(X)) # Randomly assigned coefficients (starting point)

ans <- NewtonRaphson(X, y, logisticstuff, coef) %>% data.frame() %>% `rownames<-`( NULL )
ans %>% kbl(caption = "Newton Raphson result with 5 parameters") %>%
   kable_styling(font_size = 8, latex_options = "HOLD_position")
```

Table 1: Newton Raphson result with 5 parameters

Number.of.trial	Log_likelihood	asc_dir_check	Beta0	Beta1	Beta2	Beta3	Beta4
0	-315.38197	297.4027390	0.0000000	0.000000	0.0000000	0.0000000	0.0000000
1	-141.92524	64.4620788	-8.2741164	-1.259642	0.0795466	0.2861187	-0.0034815
2	-100.99962	25.2067494	-12.8134881	-2.880231	0.1358457	0.5775260	-0.0044236
3	-85.08700	9.5366502	-13.7360236	-4.995863	0.1916826	0.8709006	-0.0001595
4	-79.07361	3.3687360	-8.6473294	-7.581528	0.2344300	1.1188065	0.0123355
5	-77.06852	0.4321716	-0.2645781	-10.005234	0.2577985	1.2831011	0.0289108
6	-76.83723	0.0057185	2.8702542	-10.980122	0.2685915	1.3503646	0.0357635
7	-76.83435	0.0000010	3.1959332	-11.094010	0.2700253	1.3588134	0.0365377
8	-76.83435	0.0000000	3.2001911	-11.095519	0.2700442	1.3589267	0.0365478
9	-76.83435	0.0000000	3.2001918	-11.095520	0.2700442	1.3589267	0.0365478

```
## Fitted glm model Beta0: 3.2 , Beta1: -11.096 , Beta2: 0.27 , Beta3: 1.359 , Beta4: 0.037
```

Newton Raphson with all 31 parameters

```
stopQuietly <- function(...) {
  blankMsg <- sprintf("\r%s\r", paste(rep(" ", getOption("width")-1L), collapse=" "));
  stop(simpleError("cannot proceed the algorithm further due to NaN values in p vector"));
}

logisticstuff <- function(X, y, beta) {
  p <- exp(X%*%beta) / (1 + exp(X%*%beta)) %>% as.vector()
  for (i in 1:length(p)) {
    if (p[i] == 1) {
      if (sum(is.na(p) == TRUE) > 0) {
        stopQuietly()
      }
      p[i] <- 1-2e-8
    }
  }
  loglik <- t(y) %*% log(p) + t(1-y)  %*% log(1-p)
  grad <- t(X) %*% (y-p)
  W <- diag(c(p*(1-p)))
  Hess <- -t(X) %*% W %*% X
  return(list(loglik = loglik, grad = grad, Hess = Hess))
}

X <- model.matrix(diagnosis~., training_df_31p)
y <- as.matrix(training_df$diagnosis)

NewtonRaphson <- function(X, y, logit_func, start, tol=1e-10, maxiter = 200) {
   i <- 0
   cur_beta <- start
   stuff <- logit_func(X, y, cur_beta)
   asc_dir_check <- -t(stuff$grad) %*% solve(stuff$Hess) %*% stuff$grad
   res <- c(i, stuff$loglik, asc_dir_check, cur_beta)
   prevloglik <- -Inf # To make sure it iterates

   while (i < maxiter && abs(stuff$loglik - prevloglik) > tol) {
     i <- i + 1
     prevloglik <- stuff$loglik
     prev_beta <- cur_beta
     cur_beta <- prev_beta - (solve(stuff$Hess) %*% stuff$grad) #update beta
     stuff <- logit_func(X, y, cur_beta) #update log likelihood, gradient, Hessia
     asc_dir_check <- -t(stuff$grad) %*% solve(stuff$Hess) %*% stuff$grad
     res <- rbind(res, c(i, stuff$loglik, asc_dir_check, cur_beta))
     colnames(res) <- c("Number_of_trial", "Log_likelihood", "asc_dir_check", paste0("Beta", 0:30))
   }
   return(res)
}

coef <- rep(0,ncol(X)) # Randomly assigned coefficients (starting point)
```

Table 2: Newton Raphson result with 31 parameters

Number_of_trial	Log_likelihood	asc_dir_check	Beta0	Beta1	Beta2	Beta3	Beta4	Beta5	Beta6	Beta7	Beta8	Beta9
0	-315.3819672	363.3863080	0.00000	0.0000000	0.0000000	0.0000000	0.0000000	0.00000	0.00000	0.000000	0.000000	0.000000
1	-105.0649679	72.6858805	-10.22033	-0.9360000	0.0252396	0.0861849	0.0023765	3.64103	-15.79377	5.645109	8.531072	-1.065732
2	-58.3133087	34.7575625	-17.49844	-0.9011090	0.0147350	0.0982375	0.0023645	11.36727	-22.73022	11.031122	12.149248	-4.667018
3	-35.6372927	18.8281790	-25.45537	-0.9217526	-0.0361649	0.1557306	0.0014547	19.33863	-26.96863	14.924187	21.161128	-10.724041
4	-23.2348067	12.0995319	-36.38389	-1.3710840	-0.0871143	0.2339244	0.0016870	24.70603	-33.12307	20.893833	37.722687	-16.051297
5	-15.0488775	10.0318418	-56.77922	-1.9465817	-0.0753320	0.2490561	0.0028944	44.85133	-49.49142	38.654349	59.749955	-22.376123
6	-8.2985550	7.2261060	-91.13019	-3.8130106	0.0045095	0.2249788	0.0152424	101.57375	-85.94669	86.558419	63.519599	-34.823661
7	-3.6072702	3.4217703	-143.73478	-6.9205335	0.1232558	0.3089762	0.0326358	173.14779	-146.55798	164.250393	55.496787	-48.981234
8	-1.4255469	1.3794906	-204.19664	-10.2275614	0.1985325	0.4651119	0.0466945	268.69662	-222.59760	241.719418	73.900583	-58.527420
9	-0.5489105	0.5370347	-265.83920	-14.7424919	0.2491523	0.7565496	0.0641260	380.75184	-305.28817	315.941712	101.668558	-67.844754
10	-0.2081442	0.2050172	-327.39849	-20.8328954	0.2917701	1.1944761	0.0883038	507.67285	-391.62166	388.866459	130.924990	-78.560275
11	-0.0781367	0.0769959	-387.87498	-28.5404951	0.3403048	1.7432844	0.1223028	643.62385	-478.60243	460.678720	158.280179	-90.801175
12	-0.0292694	0.0285672	-448.15400	-37.3858578	0.4003243	2.3821901	0.1625516	781.60031	-565.62022	530.975313	183.713482	-104.377880
13	-0.0110700	0.0105491	-511.49230	-47.1248196	0.4628012	3.2150915	0.1999765	924.99196	-656.77645	596.164400	209.470653	-120.341689
14	-0.0042974	0.0039108	-581.39421	-58.5908834	0.5219444	4.4926386	0.2274242	1080.59892	-756.77621	647.738037	236.696149	-142.884538
15	-0.0017537	0.0014675	-659.95049	-71.8607992	0.5704151	6.3975175	0.2331574	1235.62022	-866.29339	680.191582	265.579298	-176.632597
16	-0.0007773	0.0005637	-743.86695	-84.8716500	0.5851627	8.7517464	0.2063349	1360.26572	-978.98469	700.406787	291.766470	-218.051105
17	-0.0003896	0.0002290	-825.33800	-96.1825906	0.5380378	11.2189731	0.1563408	1449.67579	-1087.27638	718.014461	303.708527	-257.457592
18	-0.0002265	0.0001058	-894.49775	-104.8654488	0.4549612	13.2799599	0.1084977	1515.13021	-1178.08300	737.190130	302.491325	-288.444814
19	-0.0001500	0.0000577	-948.24398	-111.1803630	0.3844162	14.8097315	0.0717586	1564.34692	-1247.21969	755.079270	298.782277	-311.246366
20	-0.0001088	0.0000362	-989.42316	-115.9561635	0.3380406	15.9423143	0.0450139	1601.37158	-1299.24136	770.570618	296.571027	-328.551445
21	-0.0000838	0.0000251	-1021.77072	-119.7020744	0.3101736	16.8003212	0.0254486	1629.61455	-1339.56050	783.911121	296.407323	-342.244688
22	-0.0000673	0.0000187	-1048.12706	-122.7243020	0.2944686	17.4688006	0.0107932	1651.59365	-1372.07719	795.569056	297.842708	-353.426730
23	-0.0000557	0.0000147	-1070.34397	-125.2189261	0.2862967	18.0036925	-0.0004920	1669.30957	-1399.23770	805.981862	300.170991	-362.795656
24	-0.0000471	0.0000120	-1089.57958	-127.3208215	0.2827493	18.4417318	-0.0093852	1684.14261	-1422.55830	815.473086	302.916156	-370.817469
25	-0.0000405	0.0000101	-1106.57767	-129.1241741	0.2821007	18.8073906	-0.0165144	1696.96844	-1443.01125	824.263845	305.826358	-377.812617
26	-0.0000353	0.0000087	-1121.84171	-130.6944755	0.2833342	19.1173456	-0.0223029	1708.34483	-1461.24732	832.506286	308.760617	-384.007296
27	-0.0000311	0.0000077	-1135.72707	-132.0787108	0.2858292	19.3833668	-0.0270463	1718.63472	-1477.72094	840.306933	311.643563	-389.565472
28	-0.0000277	0.0000069	-1148.49342	-133.3115746	0.2891929	19.6140282	-0.0309581	1728.08208	-1492.76083	847.742274	314.436499	-394.608937
29	-0.0000249	0.0000062	-1160.33592	-134.4193253	0.2931695	19.8157464	-0.0341966	1736.85646	-1506.61166	854.868626	317.121004	-399.230130
30	-0.0000225	0.0000057	-1171.40474	-135.4222510	0.2975886	19.9934334	-0.0368822	1745.07935	-1519.45991	861.728378	319.689865	-403.500556
31	-0.0000204	0.0000053	-1181.88530	-136.3696944	0.3029300	20.1582094	-0.0392283	1752.73140	-1531.63482	868.370781	322.129645	-407.723941
32	-0.0000187	0.0000050	-1191.80581	-137.2400652	0.3081387	20.3055441	-0.0411818	1760.10577	-1543.01196	874.784604	324.400952	-411.639768
33	-0.0000171	0.0000047	-1201.23849	-138.0441882	0.3134401	20.4380993	-0.0428086	1767.14707	-1553.72447	881.004825	326.552582	-415.318482
34	-0.0000158	0.0000044	-1210.24761	-138.7906760	0.3188614	20.5579467	-0.0441587	1773.88132	-1563.85884	887.049720	328.599572	-418.802948
35	-0.0000146	0.0000042	-1218.88663	-139.4865934	0.3243979	20.6667661	-0.0452721	1780.34116	-1573.48018	892.932115	330.548954	-422.123584
36	-0.0000136	0.0000040	-1227.20008	-140.1378168	0.3300422	20.7659482	-0.0461820	1786.55470	-1582.64066	898.661724	332.406080	-425.303820
37	-0.0000127	0.0000039	-1235.22547	-140.7492787	0.3357891	20.8566603	-0.0469160	1792.54497	-1591.38343	904.245959	334.176059	-428.362488
38	-0.0000119	0.0000037	-1243.04230	-141.3264902	0.3406743	20.9406962	-0.0475476	1798.31437	-1599.85208	909.638162	335.941990	-431.408956
39	-0.0000111	0.0000036	-1250.61131	-141.8726644	0.3461432	21.0181468	-0.0480439	1803.88539	-1607.95516	914.917768	337.627142	-434.341717
40	-0.0000105	0.0000035	-1257.96833	-142.3895255	0.3518382	21.0896134	-0.0484238	1809.28189	-1615.71239	920.072476	339.226627	-437.181797
41	-0.0000099	0.0000034	-1265.13967	-142.8796968	0.3576664	21.1557643	-0.0487040	1814.51722	-1623.15545	925.099376	340.748195	-439.944133
42	-0.0000093	0.0000033	-1272.14572	-143.3456263	0.3636008	21.2171900	-0.0488987	1819.60084	-1630.31129	929.997752	342.200749	-442.639578
43	-0.0000089	0.0000032	-1279.00343	-143.7894612	0.3696339	21.2744032	-0.0490201	1824.54059	-1637.20202	934.767104	343.592625	-445.276485

```
## Error: cannot proceed the algorithm further due to NaN values in p vector
```

Table 3: Failed Newton Rapshon for 31 parameters

Result	values
Number_of_trial	42
Log_likelihood	-0.000009341111
asc_dir_check	0.000003304651
Beta0	-1272.146
Beta1	-143.3456
Beta2	0.3636008
Beta3	21.21719
Beta4	-0.04889868
Beta5	1819.601
Beta6	-1630.311
Beta7	929.9978
Beta8	342.2007
Beta9	-442.6396
Beta10	2262.871
Beta11	-28.39532
Beta12	-22.86274
Beta13	-3.83439
Beta14	2.290926
Beta15	12389.41
Beta16	5063.34
Beta17	-1606.132
Beta18	471.5062
Beta19	-5288.991
Beta20	-45056.05
Beta21	121.76
Beta22	5.032851
Beta23	-5.146386
Beta24	-0.7058646
Beta25	-923.8527
Beta26	-379.2067
Beta27	124.5956
Beta28	502.6894
Beta29	703.1726
Beta30	2613.257

Table 4: Fitted glm model coefficients of 31 parameters

names	x
(Intercept)	-1293.417
radius_mean	-172.482
texture_mean	1.066
perimeter_mean	13.475
area_mean	0.718
smoothness_mean	2785.956
compactness_mean	-1775.940
concavity_mean	1424.757
concave_points_mean	473.400
symmetry_mean	-345.545
fractal_dimension_mean	3672.362
radius_se	-254.806
texture_se	-42.316
perimeter_se	-7.104
area_se	4.773
smoothness_se	15391.485
compactness_se	5711.477
concavity_se	-2162.101
concave_points_se	4968.934
symmetry_se	-7212.775
fractal_dimension_se	-55050.387
radius_worst	178.104
texture_worst	6.587
perimeter_worst	-6.294
area_worst	-1.114
smoothness_worst	-1623.965
compactness_worst	-402.553
concavity_worst	160.429
concave_points_worst	134.278
symmetry_worst	875.003
fractal_dimension_worst	3118.667

Modified Newton Raphson with 5 number of parameters

```
logisticstuff <- function(X, y, beta) {
  p <- exp(X%*%beta) / (1+ exp(X%*%beta)) %>% as.vector()
  for (i in 1:length(p)) {
    if (p[i] == 1) {
      p[i] <- 1-1e-8
    }
  }
  loglik <- t(y) %*% log(p) + t(1-y)  %*% log(1-p)
  grad <- t(X) %*% (y-p); W <- diag(c(p*(1-p))); Hess <- -t(X) %*% W %*% X
  return(list(loglik = loglik, grad = grad, Hess = Hess))
}
X <- model.matrix(diagnosis~., training_df_5p); y <- as.matrix(training_df$diagnosis)

NewtonRaphson_mod <- function(X, y, logit_func, start, tol=1e-10, maxiter = 200) {
   i <- 0
   cur_beta <- start
   stuff <- logit_func(X, y, cur_beta)
   asc_dir_check <- -t(stuff$grad) %*% solve(stuff$Hess) %*% stuff$grad
   lambda <- 1 #initial random lambda
   res <- c(i, stuff$loglik, asc_dir_check, cur_beta)
   prevloglik <- -Inf # To make sure it iterates

   while (i < maxiter && abs(stuff$loglik - prevloglik) > tol) {
      i <- i + 1
      prev_beta <- cur_beta
      #checking if direction is ascent. If not, transform Hessian into negative definite.
      if (asc_dir_check < 0) {
         stuff$Hess = stuff$Hess - (max(stuff$Hess) + 5)
         prev_beta <- prev_beta - lambda * (solve(stuff$Hess) %*% stuff$grad)
         stuff <- logit_func(X, y, prev_beta)
         prevloglik <- stuff$loglik
      }
      else {
         prev_beta <- prev_beta - lambda * (solve(stuff$Hess) %*% stuff$grad)
         stuff <- logit_func(X, y, prev_beta)
         prevloglik <- stuff$loglik
      }
      cur2_beta <- prev_beta - lambda * (solve(stuff$Hess) %*% stuff$grad)
      stuff2 <- logit_func(X, y, cur2_beta)
      #condition check before step halving process
      if (stuff2$loglik > prevloglik) {
         cur_beta = cur2_beta
         stuff = stuff2
         asc_dir_check <- -t(stuff$grad) %*% solve(stuff$Hess) %*% stuff$grad
      }
      #step halving process
      else {
         repeat {
         lambda = lambda/2
         cur_beta = prev_beta - lambda * (solve(stuff$Hess) %*% stuff$grad)
         stuff <- logit_func(X, y, cur_beta)
```

```
        if (stuff$loglik > prevloglik) {
        cur_beta = cur_beta
        stuff = stuff
        asc_dir_check <- -t(stuff$grad) %*% solve(stuff$Hess) %*% stuff$grad
        break
        }
        }
        }
      res <- rbind(res, c(i, stuff$loglik, asc_dir_check, cur_beta))
      colnames(res) <- c("Number of trial", "Log_likelihood", "asc_dir_check", paste0("Beta", 0:4))
   }
   return(res)
}
coef <- rep(0,ncol(X)) # Randomly assigned coefficients (starting point)

ans <- NewtonRaphson_mod(X, y, logisticstuff, coef) %>% data.frame() %>% `rownames<-`( NULL )
ans %>% kbl(caption = "Newton Raphson result with 5 parameters") %>%
   kable_styling(font_size = 8, latex_options = "HOLD_position")
```

Table 5: Newton Raphson result with 5 parameters

Number.of.trial	Log_likelihood	asc_dir_check	Beta0	Beta1	Beta2	Beta3	Beta4
0	-315.38197	297.4027390	0.000000	0.000000	0.0000000	0.000000	0.0000000
1	-100.99962	25.2067494	-12.813488	-2.880231	0.1358457	0.577526	-0.0044236
2	-79.07361	3.3687360	-8.647329	-7.581528	0.2344300	1.118807	0.0123355
3	-76.83723	0.0057185	2.870254	-10.980122	0.2685915	1.350365	0.0357635
4	-76.83435	0.0000000	3.200191	-11.095519	0.2700442	1.358927	0.0365478
5	-76.83435	0.0000000	3.200192	-11.095520	0.2700442	1.358927	0.0365478

```
## Fitted glm model Beta0: 3.2 , Beta1: -11.096 , Beta2: 0.27 , Beta3: 1.359 , Beta4: 0.037
```

Modified Newton Raphson with all 31 parameters

```
logisticstuff <- function(X, y, beta) {
  p <- exp(X%*%beta) / (1 + exp(X%*%beta)) %>% as.vector()
  for (i in 1:length(p)) {
    if (p[i] == 1) {
      if (sum(is.na(p) == TRUE) > 0) {
        stopQuietly()
      }
      p[i] <- 1-2e-8
    }
  }
  loglik <- t(y) %*% log(p) + t(1-y)  %*% log(1-p)
  grad <- t(X) %*% (y-p); W <- diag(c(p*(1-p))) ; Hess <- -t(X) %*% W %*% X
  return(list(loglik = loglik, grad = grad, Hess = Hess))
}
X <- model.matrix(diagnosis~., training_df_31p); y <- as.matrix(training_df$diagnosis)

NewtonRaphson_mod <- function(X, y, logit_func, start, tol=1e-10, maxiter = 200) {
   i <- 0
```

```
   cur_beta <- start
   stuff <- logit_func(X, y, cur_beta)
   asc_dir_check <- -t(stuff$grad) %*% solve(stuff$Hess) %*% stuff$grad
   res <- c(i, stuff$loglik, asc_dir_check, cur_beta)
   prevloglik <- -Inf # To make sure it iterates
   lambda <- 1 #initial random lambda

   while (i < maxiter && abs(stuff$loglik - prevloglik) > tol) {
      i <- i + 1
      prev_beta <- cur_beta

      #checking if direction is ascent. If not, transform Hessian into negative definite.
      if (asc_dir_check < 0) {
         stuff$Hess = stuff$Hess - (max(stuff$Hess) + 5)
         prev_beta <- prev_beta - lambda * (solve(stuff$Hess) %*% stuff$grad)
         stuff <- logit_func(X, y, prev_beta)
         prevloglik <- stuff$loglik
      }
      else {
         prev_beta <- prev_beta - lambda * (solve(stuff$Hess) %*% stuff$grad)
         stuff <- logit_func(X, y, prev_beta)
         prevloglik <- stuff$loglik
      }
      cur2_beta <- prev_beta - lambda * (solve(stuff$Hess) %*% stuff$grad)
      stuff2 <- logit_func(X, y, cur2_beta)
      #condition check before step halving process
      if (stuff2$loglik > prevloglik) {
         cur_beta = cur2_beta
         stuff = stuff2
         asc_dir_check <- -t(stuff$grad) %*% solve(stuff$Hess) %*% stuff$grad
      }
      #step halving process
      else {
         repeat {
         lambda = lambda/2
         cur_beta = prev_beta - lambda * (solve(stuff$Hess) %*% stuff$grad)
         stuff <- logit_func(X, y, cur_beta)
         if (stuff$loglik > prevloglik) {
         cur_beta = cur_beta
         stuff = stuff
         asc_dir_check <- -t(stuff$grad) %*% solve(stuff$Hess) %*% stuff$grad
         break
         }
         }
         }
      res <- rbind(res, c(i, stuff$loglik, asc_dir_check, cur_beta))
      colnames(res) <- c("Number of trial", "Log_likelihood", "asc_dir_check", paste0("Beta", 0:30))
   }
   return(res)
}
coef <- rep(0,ncol(X)) # Randomly assigned coefficients (starting point)
```

Table 6: Newton Raphson result with 31 parameters

Number.of.trial	Log_likelihood	asc_dir_check	Beta0	Beta1	Beta2	Beta3	Beta4	Beta5	Beta6	Beta7	Beta8	Beta9	
0	-315.3819672	363.3863080	0.00000	0.000000	0.0000000	0.0000000	0.0000000	0.00000	0.00000	0.00000	0.00000	0.000000	
1	-58.3133087	34.7575625	-17.49844	-0.901109	0.0147350	0.0982375	0.0023645	11.36727	-22.73022	11.03112	12.14925	-4.667018	
2	-23.2348067	12.0995319	-36.38389	-1.371084	-0.0871143	0.2339244	0.0016870	24.70603	-33.12307	20.89383	37.72269	-16.051297	
3	-8.2985550	7.2261060	-91.13019	-3.813011	0.0045095	0.2249788	0.0152424	101.57375	-85.94669	86.55842	63.51960	-34.823661	1
4	-1.4255469	1.3794906	-204.19664	-10.227561	0.1985325	0.4651119	0.0466945	268.69662	-222.59760	241.71942	73.90058	-58.527420	5
5	-0.2081442	0.2050172	-327.39849	-20.832895	0.2917701	1.1944761	0.0883038	507.67285	-391.62166	388.86646	130.92499	-78.560275	9
6	-0.0292694	0.0285672	-448.15400	-37.385858	0.4003243	2.3821901	0.1625516	781.60031	-565.62022	530.97531	183.71348	-104.377880	12
7	-0.0042974	0.0039108	-581.39421	-58.590883	0.5219444	4.4926386	0.2274242	1080.59892	-756.77621	647.73804	236.69615	-142.884538	16
8	-0.0007773	0.0005637	-743.86695	-84.871650	0.5851627	8.7517464	0.2063349	1360.26572	-978.98469	700.40679	291.76647	-218.051105	18
9	-0.0002265	0.0001058	-894.49775	-104.865449	0.4549612	13.2799599	0.1084977	1515.13021	-1178.08300	737.19013	302.49132	-288.444814	18
10	-0.0001088	0.0000362	-989.42316	-115.956164	0.3380406	15.9423143	0.0450139	1601.37158	-1299.24136	770.57062	296.57103	-328.551445	19
11	-0.0000673	0.0000187	-1048.12706	-122.724302	0.2944686	17.4688006	0.0107932	1651.59365	-1372.07719	795.56906	297.84271	-353.426730	19
12	-0.0000471	0.0000120	-1089.57958	-127.320821	0.2827493	18.4417318	-0.0093852	1684.14261	-1422.55830	815.47309	302.91616	-370.817469	19
13	-0.0000353	0.0000087	-1121.84171	-130.694476	0.2833342	19.1173456	-0.0223029	1708.34483	-1461.24732	832.50629	308.76062	-384.007296	20
14	-0.0000277	0.0000069	-1148.49342	-133.311575	0.2891929	19.6140282	-0.0309581	1728.08208	-1492.76083	847.74227	314.43650	-394.608937	20
15	-0.0000225	0.0000057	-1171.40474	-135.422251	0.2975886	19.9934334	-0.0368822	1745.07935	-1519.45991	861.72838	319.68987	-403.500556	20
16	-0.0000187	0.0000050	-1191.80581	-137.240065	0.3081387	20.3055441	-0.0411818	1760.10577	-1543.01196	874.78460	324.40095	-411.639768	21
17	-0.0000158	0.0000044	-1210.24761	-138.790676	0.3188614	20.5579467	-0.0441587	1773.88132	-1563.85884	887.04972	328.59957	-418.802948	21
18	-0.0000136	0.0000040	-1227.20008	-140.137817	0.3300422	20.7659482	-0.0461820	1786.55470	-1582.64066	898.66172	332.40608	-425.303820	21
19	-0.0000119	0.0000037	-1243.04230	-141.326490	0.3406743	20.9406962	-0.0475476	1798.31437	-1599.85208	909.63816	335.94199	-431.408956	22
20	-0.0000105	0.0000035	-1257.96833	-142.389525	0.3518382	21.0896134	-0.0484238	1809.28189	-1615.71239	920.07248	339.22663	-437.181797	22
21	-0.0000093	0.0000033	-1272.14572	-143.345626	0.3636008	21.2171900	-0.0488987	1819.60084	-1630.31129	929.99775	342.20075	-442.639578	22

```
## Error: cannot proceed the algorithm further due to NaN values in p vector
```

Conundrum:

It is to be observed most of the absolute values of β_i continue to increase as Newton Raphson algorithm proceeds. This causes some of the elements in p vector to be very close to 1, leading some of the elements in log(1-p) vector to be negative infinity and hence the next log likelihood to diverge to negative infinity. As a result, Newton Raphson algorithm cannot go further till its convergence of maximum likelihood estimation.

Proof for why Newton Raphson algorithm cannot reach convergence

```
glm_model_beta_vector <- beta2[2] %>% pull()
p <- exp(X %*% glm_model_beta_vector) / (1 + exp(X %*% glm_model_beta_vector))
count <- which(p == 1) %>% length()
total <- length(p)
cat("Number of p = 1:",count,"out of",total)
```

```
## Number of p = 1: 143 out of 455
```

```
log_likelihood <- t(y) %*% log(p) + t(1-y)  %*% log(1-p)
cat("log_likelihood:",log_likelihood)
```

```
## log_likelihood: NaN
```

Logistic regression could be unstable when the classes are well separated, namely when β_i in a model is large as it is the case for our logistic model for the cancer data.