STATS 216: HW2

Xiangpeng Li

1. Suppose we collect data for a group of students in a statistics class with variables $X_1 =$ hours studied, $X_2 =$ undergrad GPA, and $Y =$ receive an A. We fit a logistic regression and produce estimated coefficient, $\hat{\beta}_0 = -5$, $\hat{\beta}_1 = 0.075$, $\hat{\beta}_2 = 1.1$.

    a. Estimate the probability that a student who studies for 40 hours and has an undergrad GPA of 3.5 gets an A in the class.

    We define the multiple linear model as

    $$log(\frac{p(X)}{1-P(X)}) = \beta_0 + \beta_1 * X_1 + \beta_2 * X_2$$

    Then we can plug in these estimate values, we can get $log(\frac{p(X)}{1-P(X)}) = -5 + 0.075 * 40 + 1.1 * 3.5 = 1.85$ and we take the logarithm on both side we get $\frac{p(X)}{1-p(X)}$ = 6.35982, so $P(X) = 0.864$

    b. How many hours would the student in part (a) need to study to have a 50% chance of getting an A in the class?

    $0.5 = \frac{e^{-5+0.075*X_1+1.1*X_2}}{1+e^{-5+0.075*X_1+1.1*X_2}}$,

    so $e^{-5+0.075*X_1+1.1*X_2} = 1, -5 + 0.075 * X_1 + 1.1 * X_2 = 0, X_2 = 3.5$ we can get $X_1 = 15.3$

2. Suppose we estimate the regression coefficients in a linear regression model by minimizing

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p}|\beta_j| \leq s$$

for a particular value of s. For parts (a) through (e), indicate which of i. through v. is correct. Justify your answer.

    a. As we increase s from 0, the training RSS will:

    Streadily decrease. As we loosen the restriction on the coefficients, the model will fit the the training set more and more which will decrease the RSS.

    b. Repeat (a) for test RSS.

    Decrease initially, and then eventually start increaseing in a U shape. First the RSS will decrease because as the model becomes more flexible and it will hit a minimal point, then the RSS will increase because model tends to overfit the training set and variance start to play an important part of the total error.

    c. Repeat (a) for variance.

    Steadily increase. According to the bias-variance trade off, A more flexible model will have a higher variance. When we increase s from 0, the flexiblity is increasing, thus the variance is increasing.

d. Repeat (a) for (squared) bias.

Steadily decrease. According to the bias-variance trade off, A more flexible model will have a lower bias. When we increase s from 0, the model is becoming more flexible, thus the bias is decreasing.

e. Repeat (a) for the irreducible error.

Remain constant. No matter how perfect our estimation is, there is still some error in it and cannot be reduced by model.

3. Suppose that we use some statistical learning method to make a prediction for the response Y for a particular value of the predictor $X = x$. Carefully describe how we might estimate a confidence interval for (a) our prediction of Y at x and (b) the $E(Y|X = x)$.

a. we can use the bootstrap. We repeatly sample observations from the original data set and each time we can compute the estimate value $\hat{\alpha}^{*B}$. After multiple times, we can get the mean of response and standard deviation. Based on these two parameters we can compute the confidence interval.

b. We random pick observations from the original data set to do the bootstrap and record the response and that specific value of predictor. After repeatint this process multiple times, we can compute the $E(Y|X = x)$ based on the records.

4. Logistic regression can give poor results when the two classes can be perfectly separated by a linear decision boundary. Consider just a logistic regression with a single predictor, X, so that our model is

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

Remember that the estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ maximize the likelihood function

$$L(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'}))$$

for the observed data $\{(x_i, y_i)\}_{i=1}^n$.

a. Show that the likelihood function $L(\beta_0, \beta_1)$ is always strictly less than 1.

Since $p(x_i)$ denotes the probability of the response of the i^{th} observation and can be calcuted through $p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$, no matter what value we give to X, it can only be between 0 and 1. The likelihood function is the product of two cumulative product, and each value is between 0 and 1, so the final result will always in range of 0 and 1.

b. Suppose that all of the x_i corresponding to $y_i = 0$ are negative, and all of the other x_i are positive. In that case, show that we can get $L(\beta_0, \beta_1)$ arbitrarily close to 1. That is, show that for any value $a < 1$, no matter how close to 1, you can always find values β_0 and β_1 for which $L(\beta_0, \beta_1) > a$. Explain why this means that $\hat{\beta}_0$ and $\hat{\beta}_1$ are undefined.

As we know from the question, $P(Y = 0|X < 0) = 1, P(Y = 1|X > 0) = 1$. Ideally based on these equations the likelihood function should equal to 1. However according to Question (a), we know the likelihood function is always strictly less than 1, so we cannot get the best estimates for $\hat{\beta}_0$ and $\hat{\beta}_1$ which has the largest likelihood value. Because there are always better estimates for $\hat{\beta}_0$ and $\hat{\beta}_1$ to make the likelihood function apporoach to 1 but never reach 1.

c. Show that $\hat{\beta}_0$ and $\hat{\beta}_1$ are similarly undefined if there is *any* value c for which every x_i corresponding to $y_i = 0$ is less than c and every other x_i is larger than c. In fact, the same is true for multivariate logistic regression. Whenever there is a linear decision boundary that perfectly separates the two classes, the maximum likelihood logistic regression coefficients are undefined (but you don't have to prove this last fact).

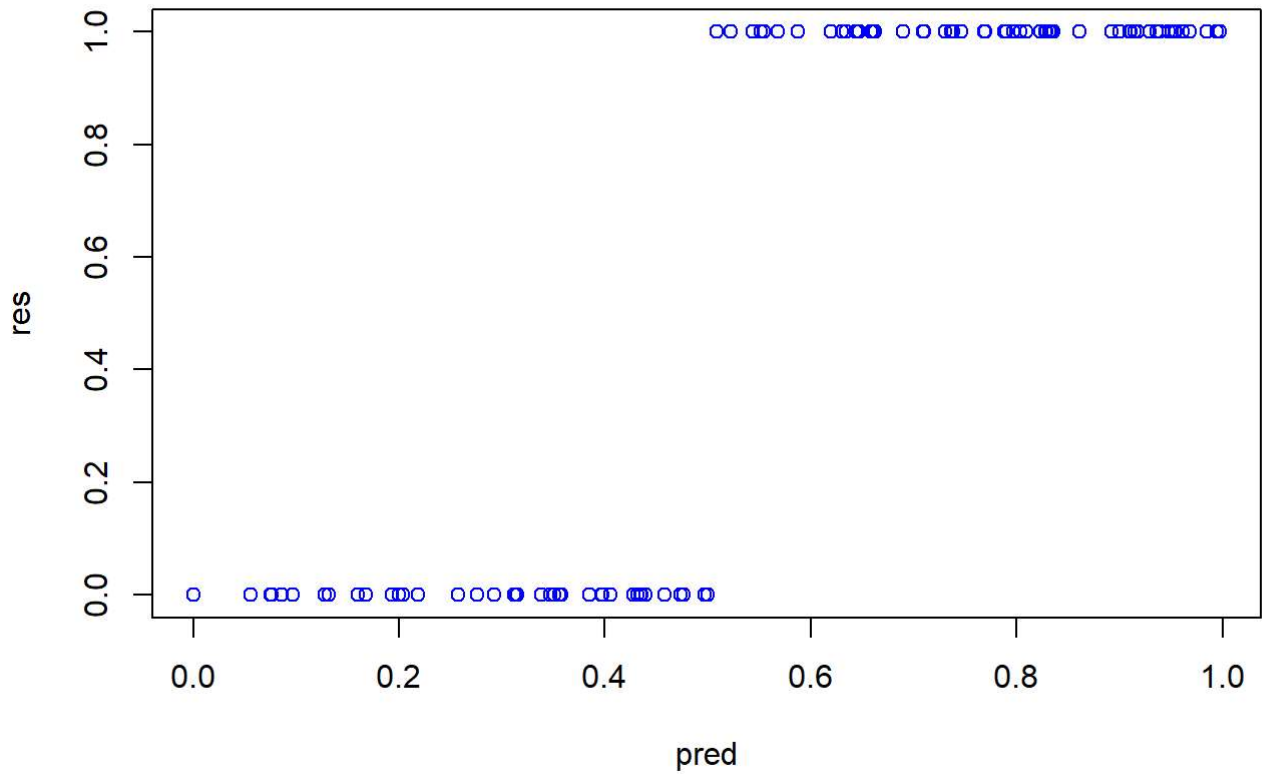
We can denote the probability as $P(Y = 0|X < c) = 1, P(Y = 1|X > c) = 1$. No matter what value c has, it's still the same concept as Question (b), we can't get the best estimates for $\hat{\beta}_0$ and $\hat{\beta}_1$ which has the largest likelihood value because the likelihood function is always apporoaching to 1 but never reach 1.

d. Come up with your own data set of the form in (c) and fit a logistic regression to it in R. Plot your data, as well as the logistic regression fit $\hat{p}(x)$.
You will probably get warning messages that the fit didnot converge, and that you have numerically 0 or 1 fitted probabilities. The first message usually signals that you have fit a logistic regression to perfectly separable classes.

```
# predictor
pred = runif(100, 0, 1)

# response
res = rep(0, 100)
res[pred > 0.5] = 1

plot(res ~ pred, col = "blue")
```



```
fit <- glm(res ~ pred, family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(fit)
```

```
##
## Call:
## glm(formula = res ~ pred, family = binomial)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -3.723e-04  -2.000e-08   2.000e-08   2.000e-08   2.700e-04
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -1914     189077   -0.01    0.992
## pred            3800     375330    0.01    0.992
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1.3606e+02  on 99  degrees of freedom
## Residual deviance: 2.6547e-07  on 98  degrees of freedom
## AIC: 4
##
## Number of Fisher Scoring iterations: 25
```

5. This question should be answered using the `Weekly` dataset, which is part of the `ISLR` package. This data is similar in nature to the `SMarket` data used in section 4.6 of our textbook, except that it contains 1,089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.

    a. Use the full dataset to perform a logistic regression with `Direction` as the `response` and the five lag variables plus `Volume` as predictors. Call your model `glm.fit` . Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so, which ones?

```
require(ISLR)
```

```
## Loading required package: ISLR
```

```
attach(Weekly)
glm.fit <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Weekly, family = bi
nomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Weekly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

$Log2$ has the relatively smallest p-value and is more statistically significant than other variables.

b. Use the following code to produce the confusion matrix for this problem.

```
glm.probs = predict(glm.fit, type = "response")
glm.pred = rep("Down", length(glm.probs))
glm.pred[glm.probs > 0.5] = "Up"
table(glm.pred, Weekly$Direction)
```

Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression.

```
glm.probs = predict(glm.fit, type = "response")
glm.pred = rep("Down", length(glm.probs))
glm.pred[glm.probs > 0.5] = "Up"
table(glm.pred, Weekly$Direction)
```

```
##
## glm.pred Down  Up
##     Down   54  48
##     Up    430 557
```

```
mean(glm.pred == Weekly$Direction)
```

```
## [1] 0.5610652
```

This matrix tell us the misclassification rate on the training data. The diagonal/anti-diagonal of the matrix is showing how many correct/wrong predictions we make on the training data.

c. Now fit the logistic regression model using a training data period from 1990 to 2008, with `Lag1`, `Lag2`, and `Lag3` as the only predictors. Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010).

```
traingSet = subset(Weekly, Year %in% 1990: 2008)
testSet = subset(Weekly, Year %in% 2009: 2010)
# fit the logitics model
glm.fit <- glm(Direction ~ Lag1 + Lag2 + Lag3, data = traingSet, family = binomial)

# predict on the test set
glm.probs = predict(glm.fit, testSet)
glm.pred = rep("Down", length(glm.probs))
glm.pred[glm.probs > 0.5] = "Up"
table(glm.pred, testSet$Direction)
```

```
##
## glm.pred Down Up
##     Down   39 56
##     Up      4  5
```

```
mean(glm.pred == testSet$Direction)
```

```
## [1] 0.4230769
```

d. Repeat (c) using LDA. Use `library(MASS)` to work with the `lda()` command.

```
library(MASS)
# fit the logitics model
lda.fit = lda(Direction ~ Lag1 + Lag2 + Lag3, data = traingSet)

# predict on the test set
lda.pred = predict(lda.fit, testSet)
lda.class = lda.pred$class
table(lda.class, testSet$Direction)
```

```
##
## lda.class Down Up
##      Down    8  9
##      Up     35 52
```

```
mean(lda.class == testSet$Direction)
```

```
## [1] 0.5769231
```

e. Repeat (c) using KNN with $K = 3$. Invoke `library(class)` to work with the `knn()` command. Set your seed to 2018 via `set.seed(2018)` .

```
library(class)
set.seed(2018)
train.X = traingSet[, c("Lag1", "Lag2", "Lag3")]
test.X = testSet[, c("Lag1", "Lag2", "Lag3")]
train.Direction = traingSet$Direction
knn.pred = knn(train.X, test.X, train.Direction, k = 3)
table(knn.pred, testSet$Direction)
```

```
##
## knn.pred Down Up
##     Down   19 27
##     Up     24 34
```

```
mean(knn.pred == testSet$Direction)
```

```
## [1] 0.5096154
```

f. Which of the models from parts (c), (d), and (e) appears to provide the best results on this data?

LDA has the best result of 0.5769231.

g. What is one scenario in which you might expect an LDA model to outperform a logistic regression model?

LDA assumes the observations are drawn from a Gaussian distribution with a common convariance matrix in each class, so if this assumeption is met, LDA will outperform a logistic regression model.

h. What is one scenario in which you might expect a KNN model to outperform a logistic regression model?

Logistic model can only generate a linear boundary, so when the decision boundary is highly non-linear, the KNN will outperform a logistic regression model.

6. **You may work in groups up to size 4 on this problem. If you do work in groups, write the names of all your group members on your problem set.**
In class, we fit a linear regression model to the NCAA data using the scores as a continuous response. y_i represented the (possibly negative) margin of victory for the home team, and (including home court advantage) we modeled:

$$y_i = \beta_0 h_i + \beta_{\text{home}(i)} - \beta_{\text{away}(i)} + \epsilon_i$$

One criticism of this model is that it may give teams too much credit for running up the score on their weaker opponents. Increasing one's margin of victory from 30 to 35 points is assigned the same importance as changing a 2-point loss to 3-point win.
To answer this criticism, we could change our model so that it only takes win-loss outcomes into account. Defining z_i now as a 0/1 indicator of whether the home team won the game, the "logistic regression version" of our model for $P(z_i = 1)$ is of the form:

$$p(\text{home}(i), \text{away}(i), h_i) = \frac{e^{\beta_0 h_i + \beta_{\text{home}(i)} - \beta_{\text{away}(i)}}}{1 + e^{\beta_0 h_i + \beta_{\text{home}(i)} - \beta_{\text{away}(i)}}}$$

Except for the home-team advantage term, this is a version of the famous *Bradley- Terry Model*, used in college football, chess, and elsewhere to compute automatic team rankings.

a. Fit the logistic regression model above to the data and examine the rankings. What happened to make both the team `saint-mary-saint-mary` and the team `st.-thomas-(tx)-celts` look so good?

```
games = read.csv("http://statweb.stanford.edu/~tibs/stats216/games.csv", as.is=T)
teams = read.csv("http://statweb.stanford.edu/~tibs/stats216/teams.csv", as.is=T)
all.teams = sort(unique(c(teams$team, games$home, games$away)))


# the response is the only difference from the previous linear regression model
y = with(games, homeScore - awayScore)
y = ifelse(y > 0, 1, 0)

X0 = as.data.frame(matrix(0, nrow(games), length(all.teams)))
names(X0) = all.teams
for(tm in all.teams) {
  X0[[tm]] = 1*(games$home == tm) - 1*(games$away == tm)
}
# set stanford as baseline
X = X0[, names(X0) != "stanford-cardinal"]

#only consider regular games
reg.season.games = which(games$gameType == "REG")

fit = glm(y ~ 0 + ., data = X, subset = reg.season.games, family = binomial)

# sort by the belta and display the first few records
head(sort(fit$coefficients, decreasing = TRUE))
```

```
## `saint-mary-saint-mary` `st.-thomas-(tx)-celts`      `gonzaga-bulldogs`
##               13.149248               12.755916                2.494628
##  `louisville-cardinals`      `indiana-hoosiers`       `kansas-jayhawks`
##                2.256516                2.193144                2.161225
```

```
# coefficients for st.-thomas-(tx)-celts and saint-mary-saint-mary
coef = summary(fit)$coefficients
head(coef[order(coef[, 4], decreasing = TRUE),])
```

```
##                                  Estimate   Std. Error      z value
## `southern-miss-golden-eagles`   0.002508263    0.6538185  0.003836330
## `st.-thomas-(tx)-celts`        12.755915722 2399.5448430  0.005315973
## `saint-mary-saint-mary`        13.149247709 2399.5448130  0.005479893
## `slippery-rock-the-rock`      -14.666488011 2399.5448185 -0.006112196
## `hanover-panthers`            -15.320750735 2399.5448120 -0.006384857
## `chadron-state-eagles`        -15.628189898 2399.5448108 -0.006512981
##                                  Pr(>|z|)
## `southern-miss-golden-eagles` 0.9969391
## `st.-thomas-(tx)-celts`       0.9957585
## `saint-mary-saint-mary`       0.9956277
## `slippery-rock-the-rock`      0.9951232
## `hanover-panthers`            0.9949057
## `chadron-state-eagles`        0.9948034
```

if we look into `games` variable we can find out that both `saint-mary-saint-mary` and `st.-thomas-(tx)-celts` only played one game and they won that only game, so there are not enough training data to fit our model which will introduce a lot of bias. Also the large standard errors and p-values for these two team indicate that logistic regression model can't perfectly describe the relationship.

```
(b) Get rid of teams that played fewer than five games and refit the model. Make a rank table li
ke the ones we made in class, where you compare the logistic regression rankings to the linear r
egression rankings, the AP Rankings, and the USA Today rankings. Which model seems to correspond
better to the voters' decisions, the linear regression or logistic regression?
```

```
# filter out teams that only play fewer than five games
game.count = table(c(games$home, games$away))
game.count = game.count[game.count >= 5]
filtered.teams = all.teams[all.teams %in% c(names(game.count))]

# set up the predictor as in question (a)
X1 = as.data.frame(matrix(0, nrow(games), length(filtered.teams)))
names(X1) = filtered.teams

for(tm in filtered.teams) {
  X1[[tm]] = 1*(games$home == tm) - 1*(games$away == tm)
}

X = X1[, names(X1) != "stanford-cardinal"]

logistic = glm(y ~ 0 + ., data = X, subset = reg.season.games, family = binomial)

y1 = with(games, homeScore - awayScore)

# calculate linear model for comparsion
linear = lm(y1 ~ 0 + ., data=X, subset=reg.season.games)


logisticCoef = coef(summary(logistic))[, 1][paste("`", teams$team, "`", sep = "")]

linearCoef = coef(summary(linear))[, 1][paste("`", teams$team, "`", sep = "")]


rank.table = cbind(
  "Logistic Fit Rank" = rank(-logisticCoef, ties="min"),
  "Linear Fit Rank" = rank(-linearCoef, ties="min"),
  "AP Rank"     = teams$apRank,
  "USAT Rank"   = teams$usaTodayRank)

# display table
rank.table[order(teams$apRank, decreasing = FALSE)[1:25], ]
```

```
##                            Logistic Fit Rank Linear Fit Rank AP Rank
## `gonzaga-bulldogs`                          1               5       1
## `louisville-cardinals`                      2               3       2
## `kansas-jayhawks`                           4               6       3
## `indiana-hoosiers`                          3               1       4
## `miami-(fl)-hurricanes`                    11              13       5
## `duke-blue-devils`                          6               4       6
## `ohio-state-buckeyes`                       5               7       7
## `georgetown-hoyas`                          7              14       8
## `michigan-state-spartans`                   9              12       9
## `michigan-wolverines`                       8               8      10
## `new-mexico-lobos`                         10              29      11
## `kansas-state-wildcats`                    12              33      12
## `saint-louis-billikens`                    14              22      13
## `florida-gators`                           22               2      14
## `marquette-golden-eagles`                  15              26      15
## `syracuse-orange`                          13              10      16
## `oklahoma-state-cowboys`                   19              17      17
## `wisconsin-badgers`                        17              11      18
## `memphis-tigers`                           16              40      19
## `pittsburgh-panthers`                      20               9      20
## `arizona-wildcats`                         28              34      21
## `creighton-bluejays`                       33              15      22
## `notre-dame-fighting-irish`                21              24      23
## `ucla-bruins`                              38              55      24
## `oregon-ducks`                             46              61      25
##                            USAT Rank
## `gonzaga-bulldogs`                 1
## `louisville-cardinals`             2
## `kansas-jayhawks`                  3
## `indiana-hoosiers`                 5
## `miami-(fl)-hurricanes`            4
## `duke-blue-devils`                 7
## `ohio-state-buckeyes`              6
## `georgetown-hoyas`                 8
## `michigan-state-spartans`          9
## `michigan-wolverines`             11
## `new-mexico-lobos`                10
## `kansas-state-wildcats`           14
## `saint-louis-billikens`           13
## `florida-gators`                  12
## `marquette-golden-eagles`         16
## `syracuse-orange`                 18
## `oklahoma-state-cowboys`          19
## `wisconsin-badgers`               17
## `memphis-tigers`                  15
## `pittsburgh-panthers`             22
## `arizona-wildcats`                20
## `creighton-bluejays`              21
## `notre-dame-fighting-irish`       NA
## `ucla-bruins`                     NA
## `oregon-ducks`                    24
```

As we can see from the above table, logistic model rank look more similiar as AP rank and USAT rank than the linear rank. We can see that logistic regression in this sceanrio fits better than the linear model.

```
(c) When we ignore the actual value of $y_i$ and instead only use whether $y_i > 0$, we are disc
arding information, so we might expect our model standard errors to be larger relative to the ef
fect sizes. If we use the linear regression model, for what fraction of teams are we confident
($p < 0.05$) that the team is better (or worse) than Stanford? For what fraction are we confiden
t if we instead use the logistic regression model?
```

```
# logistic model
a = coef(summary(logistic))
length(a[a[, 4] < 0.05, 4]) / length(a[, 4])
```

```
## [1] 0.6849711
```

```
# linear model
b = coef(summary(linear))
length(b[b[, 4] < 0.05, 4]) / length(b[, 4])
```

```
## [1] 0.9306358
```

Since we are using Stanford as baseline, all the estimate β are relative to Stanford and the Stanford β is 0 by definition. We can directly calculate the number of coeffieient which p-value is smaller than 0.05 in each model. The results are 0.68 and 0.93 for logistic model and linear model respectively.

```
(d) Each model makes a prediction about which team will win any given game. We can use ten-fold
cross-validation to estimate the test error rate for these predictions and also try to determine
whether one model is better than the other.
For each game in a given test set, there are four possible outcomes: both models are right in th
eir prediction, both are wrong, only logistic regression is right, or only linear regression is
right. Make a $2Ã□2$ contingency table displaying how often each of these four outcomes occur, o
ver all the test examples in all ten folds of cross-validation. Your table should look like:
```

	logistic right	logistic wrong
linear right	n_{11}	n_{12}
linear wrong	n_{21}	n_{22}

```
X = X[sample(nrow(X)), ]
folds = cut(seq(1, nrow(X)), breaks=10, labels=FALSE)
n11 = 0
n21 = 0
n12 = 0
n22 = 0
# linear fit
for(i in 1:10) {
  testIndexes = which(folds == i, arr.ind=TRUE)
  testData = X[testIndexes, ]
  trainData = X[-testIndexes, ]


  lm = lm(y[-testIndexes] ~ 0 + ., data = trainData, subset = reg.season.games)
  lg = glm(y[-testIndexes] ~ 0 + .,
                data = trainData,
                subset = reg.season.games,
                family = binomial)
  pred.lm = predict(lm, newdata = testData, subset = reg.season.games, type = "response")
  pred.lg = predict(lg, newdata = testData, subset = reg.season.games, type = "response")

  # record the accuracy
  for (j in i: length(testIndexes)) {

    # threshold for linear model for a winner team is 0
    if (y1[testIndexes[j]] > 0 && pred.lm[j] > 0) {
      n11 = n11 + 1
    } else {
      n21 = n21 + 1
    }

    # threshold for logistic model for a winner team is 0.5
    if (y[testIndexes[j]] > 0.5 && pred.lg[j] > 0.5) {
      n12 = n12 + 1
    } else {
      n22 = n22 + 1
    }
  }
}

result = matrix(c(n11, n12, n21, n22), ncol = 2, byrow = TRUE)
colnames(result) = c("logistic right", "logistic wrong")
rownames(result) = c("linear right","linear wrong")
result = as.table(result)
result
```

```
##              logistic right logistic wrong
## linear right           1808           1831
## linear wrong           3688           3665
```

e. We want to compare the two models to see if one is superior. n_{11} and n_{22} don't tell us anything about which model is better, because they correspond to games where both models agree with each other. So to compare the two models, we need to look at n_{12} and n_{21}. Let $D = n_{12} + n_{21}$ be the number of test games in which the two models *disagreed*.
If both models are equally good and the test set games are independent, then every time the models disagree, each model is equally likely to be right. Then, conditional on D,

$$n_{12} \ \text{Binom}(D, 1/2)$$

For large D, the above binomial distribution is approximately normal with mean $D/2$ and variance $D/4$ (hence standard deviation $\sqrt{D}/2$). **You do not have to prove any of the above statements, just take them as given.**
Use the normal approximation to carry out a test of the hypothesis that both models are equally good at predicting games. What is the conclusion of your test?
What you just did is the normal approximation to *McNemar's Test*.

```
I'm a SCPD student, don't have enought time to do this, sorry :/
```