HW3

Anish Mohan

October 14, 2015

1. Q1

- Q1a: For p=1, on an average we will be able to use 10% of the observations.
- Q1b: For p=2, we will be able to use $(0.1)^2/Area$. That is equivalent to 0.01/1=1% of the observations
- Q1c. For p=100, we will be able to use $(0.1)^{100}/Area$. That is equivalent to $(0.1^{100}/(1*1) = (10)^{-100}$=$(10)^{-98}$% of available observations
- Q1d. As shown with p=1,2 and 100, as # of features/dimensions increase, the # of available observations in the immediate vicinity of the points decrease. This decrease is exponential in nature. Hence, we find that neighbors in higher dimensions are more spread-out, therefore impacting the results we get from K-Nearest Neighbor (KNN) algorithm.
- Q1e To ensure that we get 10% of the observations for
  - p=1: We will need 10% of the area i.e 0.5 units on both sides of the point.
  - p=2: We have $s^2 = 0.1$, hence s=side of hypercube=$\sqrt{0.1} = 0.3$ i.e we need each side to be 30% of a unit square to capture 10% of the data
  - p=100: We have $s^{100} = 0.1$, hence side of hypercube is 0.977 ie. we need each side to to be 97.7% of the hypercube that contains data to capture 10% of the data i.e almost the entire dataspace has to be selected to get 10% of the uniformly distributed data.
  - As the dimensions/feature space increase, the concept of the nearest neighbor gets muddled. As in the case of 100 dimensional hypercube, we had to span almost the entire dataspace to just get 10% of the point. These 10% of nearest neighbors and are not near anymore.

2. Q2

- Q2a.

  $P(x) = \frac{e^{\beta_0+\beta_1*X_1+\beta_2*X_2}}{(1+e^{\beta_0+\beta_1*X_1+\beta_2*X_2})} = \frac{e^(-6+0.05*40+1*3.5)}{(1+e^(-6+0.05*40+1*3.5))}$

```
exp(-6+0.05*40+1*3.5)/(1+exp(-6+0.05*40+1*3.5))
```

```
## [1] 0.3775407
```

  Probability of A=37.7%

- Q2b.

  $P(x) = 0.5, X_2 = 3.5, x_1 =?$

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

  $X_1 = \frac{log(1)-\beta 0+\beta_2*X_2}{\beta_1}$

```r
(log(1)+6-3.5)/0.05
```

```
## [1] 50
```

Student has to study 50 hours to have a 50% probability of getting an A.

3. Q3

- Choose logistic regression
- KNN with K=1 has a 0 error in the training set, hence the all the errors were probably reported on the test data set. Hence error on test =36%
- Logistic Regression has a lower error in the test set and does not have a problem of overfitting in this example.

4. Q4

- 4a.

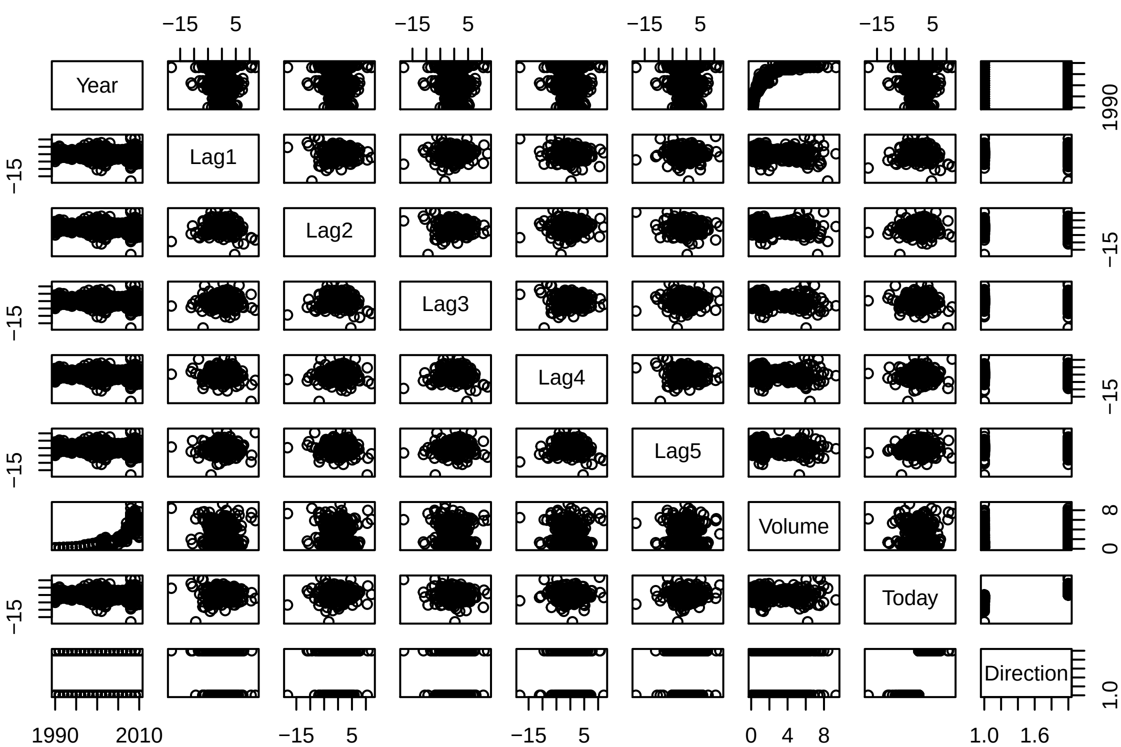
```r
library(ISLR)
weekly=Weekly
attach(weekly)
summary(weekly)
```

```
##       Year           Lag1               Lag2               Lag3
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
##  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472
##  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090
##  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260
##       Lag4               Lag5              Volume
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202
##  Median :  0.2380   Median :  0.2340   Median :1.00268
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821
##      Today          Direction
##  Min.   :-18.1950   Down:484
##  1st Qu.: -1.1540   Up  :605
##  Median :  0.2410
##  Mean   :  0.1499
##  3rd Qu.:  1.4050
##  Max.   : 12.0260
```

```r
cor(weekly[,-9])
```

```
##                Year        Lag1        Lag2        Lag3        Lag4
## Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
## Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
## Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
## Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
## Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
## Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
## Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
## Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873
##                Lag5      Volume        Today
## Year   -0.030519101  0.84194162 -0.032459894
## Lag1   -0.008183096 -0.06495131 -0.075031842
## Lag2   -0.072499482 -0.08551314  0.059166717
## Lag3    0.060657175 -0.06928771 -0.071243639
## Lag4   -0.075675027 -0.06107462 -0.007825873
## Lag5    1.000000000 -0.05851741  0.011012698
## Volume -0.058517414  1.00000000 -0.033077783
## Today   0.011012698 -0.03307778  1.000000000
```

```r
pairs(weekly)
```



- There are hardly any correlations between the variables. Year and Volume are the only variables that seem to have some significan correlation

- 4b.

```r
log_reg=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data=weekly,family=binomial)
summary(log_reg)
```

```
##
```

```
## 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
```

  - Only Lag2 has results below the p-value of 0.05 and hence appears to be statistically significant

- 4c.

```r
log_reg_prob=predict(log_reg, type="response")
log_reg_pred=rep("Down",1089)
log_reg_pred[log_reg_prob>0.5]="Up"
table(log_reg_pred,Direction)
```

```
##             Direction
## log_reg_pred Down  Up
##         Down   54  48
##         Up    430 557
```

  - From the confusion matrix:
  - ~56% ((54+557)/1089) of time the model predicts the output correctly
  - Model has significant prediction errors when the Direction is going Dow.. Of the 484 times, the direction was down, the model could only predict it correctly 54/484~11%. There was 89% prediction error.
  - When the direction was up, the model could predict with a reasonable accuracy of 92%.
  - This model overestimates when the market is going the down direction, but does well when the market is in the up direction.

- 4d.

```r
train=(Year<2009)
weekly.2009=weekly[!train,]
Direction.2009=Direction[!train]

limlog_reg=glm(Direction~Lag2,data=weekly, family=binomial,subset=train)
limlog_reg_probs=predict(limlog_reg,weekly.2009,type="response")
limlog_reg_pred=rep("Down",104)
limlog_reg_pred[limlog_reg_probs>0.50]="Up"
table(limlog_reg_pred,Direction.2009)
```

```
##                 Direction.2009
## limlog_reg_pred Down Up
##            Down    9  5
##            Up     34 56
```

  - 62.5% ((56+9)/104) predictions were made correctly.

- 4e.

```r
library(MASS)
limlda_fit=lda(Direction~Lag2,data=weekly,subset=train)
limlda_fit
```

```
## Call:
## lda(Direction ~ Lag2, data = weekly, subset = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
##
## Coefficients of linear discriminants:
##            LD1
## Lag2 0.4414162
```

```r
limlda_pred=predict(limlda_fit,weekly.2009)
limlda_class=limlda_pred$class
table(limlda_class,Direction.2009)
```

```
##             Direction.2009
## limlda_class Down Up
##         Down    9  5
##         Up     34 56
```

  - 62.5% ((56+9)/104) predictions were made correctly.

- 4f.

```r
limqda_fit=qda(Direction~Lag2,data=weekly,subset=train)
limqda_fit
```

```
## Call:
## qda(Direction ~ Lag2, data = weekly, subset = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
```

```r
limqda_class=predict(limqda_fit,weekly.2009)$class
table(limqda_class,Direction.2009)
```

```
##             Direction.2009
## limqda_class Down Up
##         Down    0  0
##         Up     43 61
```

   - 58.6% (61/104) predictions were made correctly.

- 4g.

```r
library(class)
train.X=cbind(Lag1)[train,]
test.X=cbind(Lag1)[!train,]
train.Direction=Direction[train]
set.seed(1)
knn.pred=knn(data.frame(train.X), data.frame(test.X), train.Direction, k=1)
table(knn.pred,Direction.2009)
```

```
##         Direction.2009
## knn.pred Down Up
##     Down   17 31
##     Up     26 30
```

   - 45.2% (47/104) predictions were made correctly.

- 4h Logistic Regression and Linear Discriminant Analysis provides the best results for this dataset.
- 4i
   - Did the following experiments:
      * Logistic Regression and LDA
      * Tried various combination of input predictors
      * Changed the value of threshold = 0.5 for marking 'Direction as Up'
   - QDA
      * Tried various combination of input predictors

- KNN
    - Tried with different number of neighbors.
- Best result continues to be LDA with 0.5 threshold for the probability.

```
table(limlda_class,Direction.2009)
```

```
##              Direction.2009
## limlda_class Down Up
##         Down    9  5
##         Up     34 56
```

5. **Q5**

- 5.5 Added following code to ScrapeRoster function
  weightCol = row.find('td',attrs={'class':'weight'}) if weightCol==None: entry['weight'] = np.nan else: entry['weight'] = weightCol.contents[0].strip()

Modified rosters.csv is created from python

```
athletes=read.csv("rosters.csv")
```

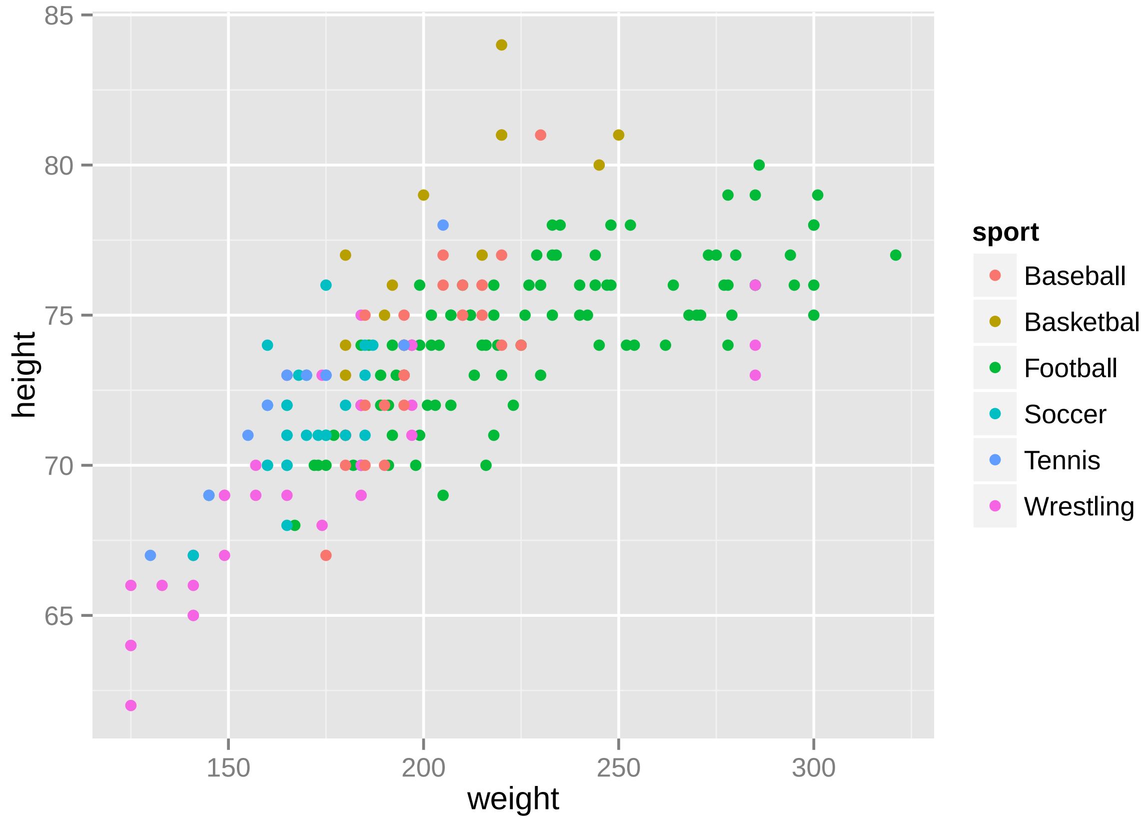
- 5.6

```
library(nnet)
library(ggplot2)
attach(athletes)
athlete.fit=multinom(sport~height+weight)
```

```
## # weights:  24 (15 variable)
## initial  value 365.518932
## iter  10 value 266.164885
## iter  20 value 207.817029
## iter  30 value 200.902009
## iter  40 value 200.198801
## iter  50 value 199.808004
## iter  60 value 199.680438
## iter  70 value 199.653822
## iter  80 value 199.647074
## iter  90 value 199.645164
## final  value 199.644727
## converged
```

- 5.7

```
qplot(weight,height,colour=sport)
```



- 5.8

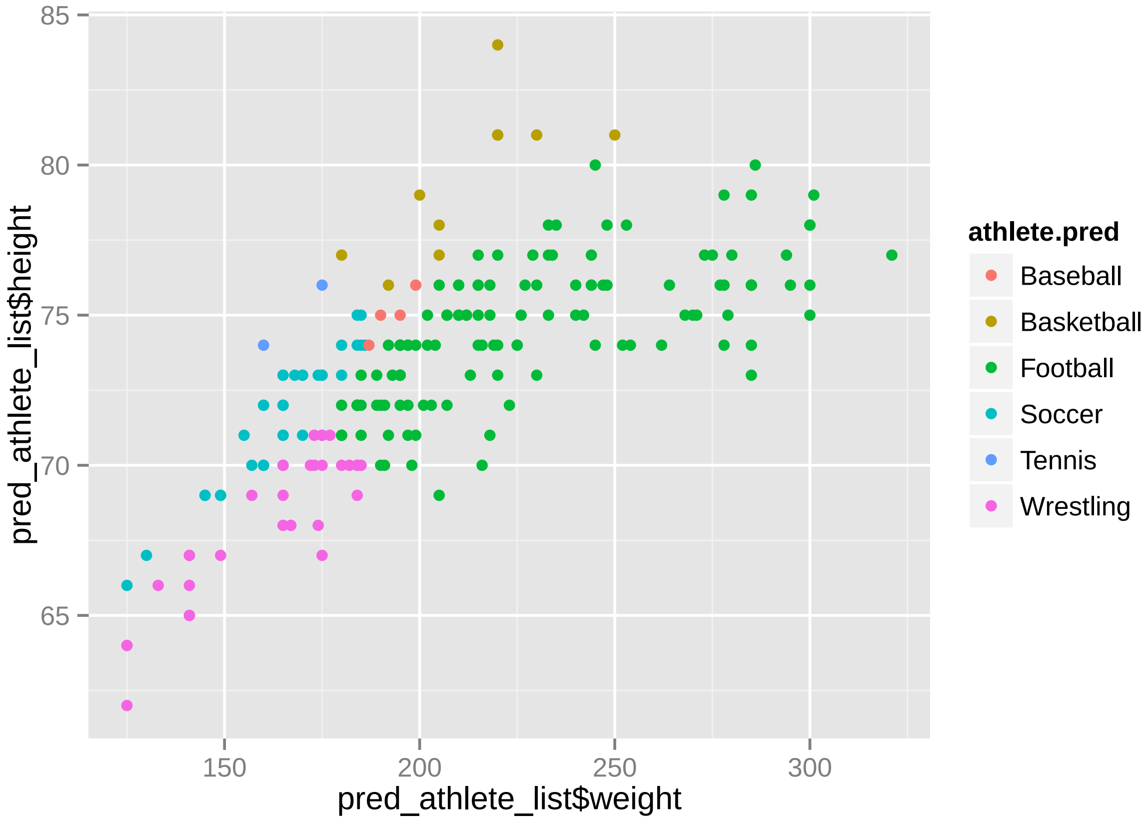
```
athlete.pred=predict(athlete.fit,athletes["sport"])
```

- 5.9

```
var=c(3,7)
pred_athlete_list=athletes[,var]
```

```
cbind(pred_athlete_list,athlete.pred)
```

```
qplot(pred_athlete_list$weight,pred_athlete_list$height,colour=athlete.pred)
```



- 5.10

```r
athlete_actual=unlist(athletes["sport"])
athlete_predicted=unlist(athlete.pred)
table(athlete_predicted,athlete_actual)
```

```
##                  athlete_actual
## athlete_predicted Baseball Basketball  Football Soccer Tennis Wrestling
##        Baseball          1          1         1      1      0         0
##        Basketball        2          7         0      0      1         0
##        Football         22          2        84      4      1        12
##        Soccer            1          2         2     12      8         7
##        Tennis            0          0         0      2      0         0
##        Wrestling         3          0         8      6      0        14
```

```
+  Based on the confusion matrix, here are the error rates
  +   Baseball =  0.75 (1-1/4)
  +   Basketball = 0.3 (1-7/10)
  +   Football = 0.328 (1-84/125)
  +   Soccer= 0.625 (1-12/32)
  +   Tennis = 1 (1-0/2)
  +   Wrestling = 0.548 (1-14/31)
```