

DSCI445 - Homework 2

Reza Gouklaney

Be sure to set `.seed(445)` at the beginning of your homework.

```
#reproducibility
set.seed(445)
```

Regression

In this exercise you will create some simulated data and will fit simple linear regression models to it. Make sure to use `set.seed(445)` prior to starting part (a) to ensure reproducible results.

- (a) Using the `rnorm()` function, create a vector x containing 100 observations drawn from a $N(0,1)$ distribution. This represents the feature, X .

```
x <- rnorm(100, 0, 1)
eps <- rnorm(100, 0, 0.25)
y <- -1 + 0.5*x + eps
length(y)

## [1] 100
```

- (b) Using the `rnorm()` function, create a vector containing 100 observations drawn from a $N(0,0.25)$ distribution, i.e. a Normal distribution with mean zero and variance 0.25.

```
eps <- rnorm(100, 0, 0.25)
```

- (c) Using x and eps , generate a vector y according to the model

What is the length of the vector y ? What are the values of β_0 and β_1 in this linear model? length of y is 100 and $\beta_0=-1$ and $\beta_1=0.5$ $\text{length}(y)=100$ $\beta_0=-1$ $\beta_1=0.5$

```
y <- -1 + 0.5*x + eps
length(y)

## [1] 100
```

- (d) Create a scatterplot displaying the relationship between x and y . Comment on what you observe.

The scatterplot shows a positive linear relationship between x and y with some random noise around the line.

```
````r
plot(x, y, main = "Scatterplot of x and y", xlab = "x", ylab = "y")
````
```

```
<!-- -->
```

- (e) Fit a least squares linear model to predict y from x . Comment on the model obtained. How do $\hat{\beta}_0$ and $\hat{\beta}_1$ compare to β_0 and β_1 ?

The estimated intercept coefficient is -0.9838296 and the estimated coefficient for x is 0.5041202 so we can say that comparing these estimated coefficients to the true coefficients (β_0 and β_1) we can see that they are close and this suggests that the least squares linear model is able to capture the underlying relationship between x and y reasonably well. The estimated intercept is close to the true intercept of -1 and the estimated coefficient for x slope is close to the true coefficient of 0.5.

```
```r
lm_model <- lm(y ~ x)

Print the coefficients of the linear model
coefficients(lm_model)
```

```
(Intercept) x
-0.9838296 0.5041202
```
```

- (f) Display the least squares line on the scatterplot obtained in (d) in blue. Draw the population regression line on the plot in red. (See `geom_abline()` for how to add a line based on intercept and slope.)

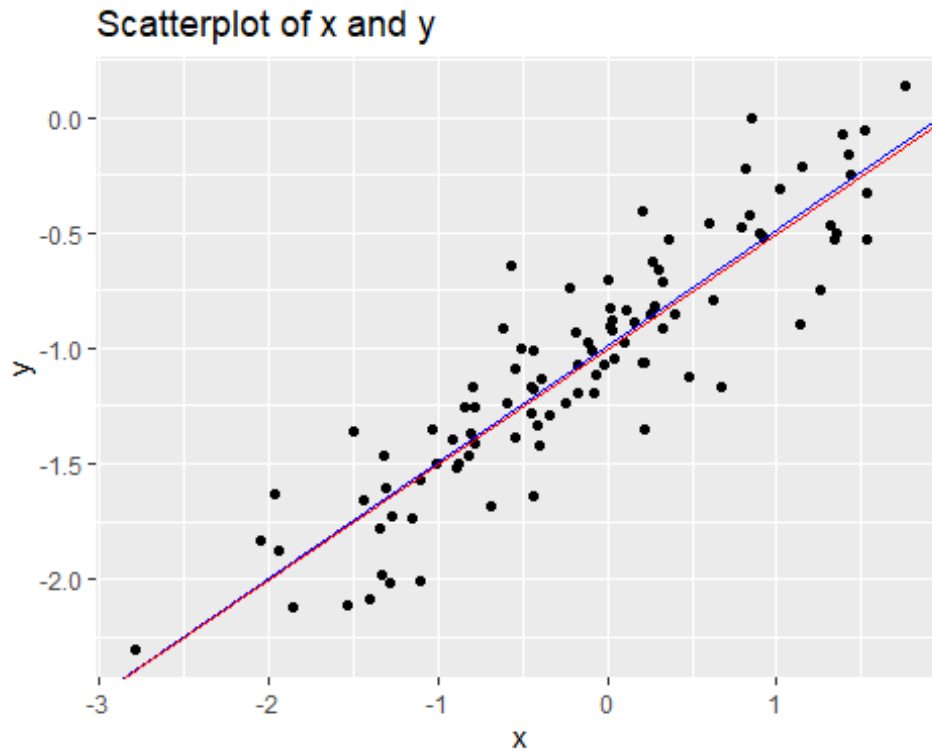
```
library(ggplot2)
data <- data.frame(x = x, y = y)
beta0 <- -1
beta1 <- 0.5

# Scatterplot
scatterplot <- ggplot(data, aes(x, y)) +
  geom_point() +
  labs(title = "Scatterplot of x and y", x = "x", y = "y")

# Blue LS line
scatterplot_ls <- scatterplot +
  geom_abline(intercept = coef(lm_model)[1], slope = coef(lm_model)[2],
    color = "blue")

# population regression line shown in red
scatterplot_ls_population <- scatterplot_ls +
  geom_abline(intercept = beta0, slope = beta1, color = "red")

scatterplot_ls_population
```



- (g) Now fit a polynomial regression model that predicts y using x and x^2 . Is there evidence that the quadratic term improves the model fit? Explain your answer.

We can evaluate the overall model fit using metrics such as the R-squared value or the residual sum of squares (RSS).

If the R-squared value for the polynomial model is higher, it suggests that the quadratic term improves the model fit. The R-squared value for the simple linear model is approximately 0.811682 and the R-squared value for the polynomial regression model is approximately 0.8117042.

Comparing the R-squared values, we can see that both models have fairly similar R-squared values. The polynomial regression model with the quadratic term (x^2) only slightly improves the model fit compared to the simple linear model. But the improvement is relatively small and insignificant.

```
```r
Fit a polynomial regression model
poly_model <- lm(y ~ x + I(x^2))

Print the coefficients of the polynomial model
coefficients(poly_model)
```

## (Intercept)          x          I(x^2)
```

```
## -0.98194555  0.50328330 -0.00219486
```

```r
# R-squared for the simple linear model
r_squared_linear <- summary(lm_model)$r.squared

# R-squared for the polynomial regression model
r_squared_poly <- summary(poly_model)$r.squared

# Print the R-squared values
r_squared_linear
```

```
## [1] 0.811682
```

```r
r_squared_poly
```

```
## [1] 0.8117042
```
```

- (h) Repeat (a)-(f) after modifying the data generation process in such a way that there is *less* noise in the data. The model should remain the same. You can accomplish this by changing the variance of the normal distribution used to generate the error term in (b). Describe your results.

We reduce the variance ( $\sigma$ ) in Step (b) to decrease the noise in the data. (a), (c), (d), and (e) remain the same as before.

After running this code we see a scatterplot with reduced noise where the data points are closer to the regression line. The least squares line (blue) and the population regression line (red) are also closer to each other on the scatterplot.

```
```r
sigma <- 0.1 # Reduced variance for less noise

epsilon <- rnorm(100, 0, sigma)

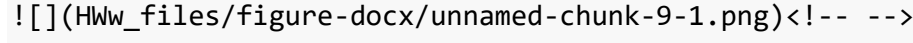
# (c): Calculating y values using the true model
y1 <- beta0 + beta1 * x + epsilon

# (d): Creating a scatterplot
plot(x, y1, main = "Scatterplot Reduced variance", xlab = "x", ylab = "y")
```

```

# (e): Fit LS linear model for prediction
lm_model_clear <- lm(y1 ~ x)

# (f): Displaying LS line and population regression line
abline(lm_model, col = "blue") # BLUE LS line
abline(a = beta0, b = beta1, col = "red") # RED Population Reg
```



```

<!-- -->

```r
coefficients(lm_model_clear)
```

## (Intercept)          x
## -1.0171805    0.4994107
```

```


```

- (i) Repeat (a)-(f) after modifying the data generation process in such a way that there is *more* noise in the data. The model should remain the same. You can accomplish this by changing the variance of the normal distribution used to generate the error term in (b). Describe your results.

This time, we increased the variance (sigma) in Step (b) to decrease the noise in the data. The rest of the steps (a), (c), (d), and (e) remain the same as before.

After running this code we see a scatterplot with reduced noise where the data points are far away from the regression line looking very non-linearly correlated or uncorrelated. The least squares line (blue) and the population regression line (red) are distinguishable from each other.

```

```r
sigma <- 2 # Increased variance for less noise

(b)
epsilon <- rnorm(100, 0, sigma)

(c)
y2 <- beta0 + beta1 * x + epsilon

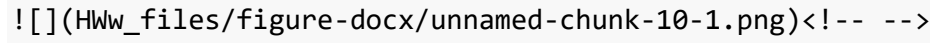
(d)
plot(x, y2, main = "Scatterplot Increased variance ", xlab = "x", ylab = "y")

(e)
lm_model_noisy <- lm(y2 ~ x)

(f)
abline(lm_model_noisy, col = "blue")

```

```
abline(a = beta0, b = beta1, col = "red")
```



```
<!-- -->

```r
coefficients(lm_model_noisy)
```

```
## (Intercept)          x
## -0.9200858    0.7349173
```
```


```

- (j) What are the confidence intervals for β_0 and β_1 based on the original data set, the noisier data set, and the less noisy data sets? Comment on your results.

For both β_0 and β_1 , the confidence intervals based on the less noisy data set are narrower compared to the original and noisier data sets. This indicates that the estimates of the coefficients are more precise and have lower uncertainty when the noise is reduced. The confidence intervals for β_0 and β_1 based on the original and noisier data sets have some overlap with each other, suggesting that the noise in the data affects the precision and accuracy of the coefficient estimates. The confidence intervals for β_0 and β_1 based on the less noisy data set are relatively narrower, indicating higher precision and more reliable estimates of the coefficients. So we can conclude that reducing the noise in the data leads to more precise coefficient estimates, as indicated by the narrower confidence intervals. This suggests that with less noise, we have more confidence in the estimated values of β_0 and β_1 .

```
```r
confint(lm_model)
```

```
2.5 % 97.5 %
(Intercept) -1.0304871 -0.9371722
x 0.4554439 0.5527965
```

```r
confint(lm_model_clear)
```

```
2.5 % 97.5 %
(Intercept) -1.0361738 -0.9981871
x 0.4795955 0.5192260
```
```

```

```r
confint(lm_model_noisy)
```

##
##           2.5 %      97.5 %
## (Intercept) -1.3296980 -0.5104735
## x           0.3075806  1.1622541
##

# simulated regression task

```

Classification

1. When the number of features p is large, there tends to be a deterioration in the performance of KNN and other *local* approaches that perform prediction using only observations that are *near* the test observation for which a prediction must be made. This phenomenon is known as the *curse of dimensionality*, and it ties into the fact that non-parametric approaches often perform poorly when p is large. We will now investigate this curse.

- (a) Suppose that we have a set of observations, each with measurements on $p = 1$ feature, X . We assume that X is uniformly (evenly) distributed on $[0,1]$. Associated with each observation is a response value. Suppose that we wish to predict a test observation's response using only observations that are within 10% of the range of X closest to that test observation. For instance, in order to predict the response for a test observation with $X = 0.6$ we will use observations in the range $[0.55, 0.65]$. On average, what fraction of the available observations will we use to make this prediction? 9.75%

When x is in range of $[0.05, 0.95]$ samples are in: $[x-0.05, x+0.05]$ thus they have $(x+0.05) - (x-0.05) = 0.1$ or 10% length.

But when $x < 0.05$ observations are in: $[0, x+0.05]$ which has $x+0.05-0$ or $(100x+5)$ % length.

when $x > 0.95$ the length is $(105-100x)\%$ And considering the whole $[0,1]$ interval and integrating over all three sub intervals:

$$\text{Int}(10) \text{ over } [0.05, 0.95] + \text{Int}(100x+5) \text{ over } [0, 0.05] + \text{Int}(105-100x) \text{ over } [0.95, 1] = 9.75$$

And the final answer is 9.75% .

- (b) Now suppose that we have a set of observations, each with measurements on $p = 2$ features, X_1 and X_2 . We assume (X_1, X_2) are uniformly distributed on $[0,1] \times [0,1]$. we wish to predict a test observation's response using only observations that are within 10% of the range of X_1 and within 10% of the range of X_2 closest to that test observation. For instance, in order to predict the response for a test observation with $X_1 = 0.6$ and $X_2 = 0.35$ we will use observations in the range $[0.55, 0.65]$ for X_1 and in the range $[0.3, 0.4]$ for X_2 . On average, what fraction of the available observations will we use to make this prediction?

Assuming these two features are independent, $p = 0.0975 * 0.0975 = 0.00950625$ so the fraction of the available observations is 0.950625 %

(c) Now suppose that we have a set of observations, each with measurements on $p = 100$ features. Again, the observations are uniformly distributed on each feature, and again each feature ranges from 0 to 1. We wish to predict a test observation's response using only observations that are within 10% of each feature's range that is closest to that test observation. What fraction of the available observations will we use to make this prediction?

Assuming the independence of the 100 observations from each other the fraction of the available observations is: $(0.0975)^{100} = 7.95 * (10)^{-102}$ which is approximately equal to ZERO.
So, the final answer is about 0%

(d) Using your answers to (a)--(c), argue that a drawback of KNN when p is large is that there are very few training observations *near* any given test observation.

Since when we have p number of features, the fraction of available observations we will use to make the prediction is $(0.0975)^p$, one drawback of KNN is that when p is too large, there are very few observations near any given test observation.

Note that as p goes to infinity, $(0.0975)^p$ approaches to zero as we saw in part (c)

(e) Now suppose that we wish to make a prediction for a test observation by creating a p -dimensional hypercube centered around the test observation that contains, on average, 10% of the training observations. For $p = 1, 2,$ and 100 , what is the length of each side of the hypercube? Comment on your answer.

[Hint: A hypercube is a generalization of a cube to an arbitrary number of dimensions. When $p = 1$, a hypercube is simply a line segment. When $p = 2$ it is a square, and when $p = 100$ it is a 100-dimensional cube.]

Generally for any number p length of each side will be $(0.1)^{(1/p)}$

So

when $p=1$: length of each side of the hypercube = 0.1

when $p=2$: length of each side of the hypercube = $0.1^{(1/2)}$

when $p=100$: length of each side of the hypercube = $0.1^{(1/100)}$

- 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 coefficients, $\hat{\beta}_0 = -6, \hat{\beta}_1 = 0.05, \widehat{\beta}_2 = 1$.

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

inserting the given values into the logistic regression formula we have: $p = (\exp(-6 + 0.05X_1 + X_2)) / (1 + \exp(-6 + 0.05X_1 + X_2)) = 0.3775$

- (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?

$p = (\exp(-6 + 0.05X_1 + 3.5)) / (1 + \exp(-6 + 0.05X_1 + 3.5)) = 0.5$ resulting in $X_1 = 50$ so final answer is 50 hours.

3. This question should be answered using the Weekly data set, which is part of the ISLR package. This data contains weekly percentage returns for the S&P 500 stock index between 1990 and 2010.

- (a) Produce some numerical and graphical summaries of the Weekly data. Do there appear to be any patterns?

After running the codes, the correlations between the lag variables and today's returns are close to zero. The only substantial correlation is between Year and Volume and when we plot Volume we see that it is increasing over time. Also pair-wise plots shows that there aren't any clear relationships between any two variable except for Year and Volume and maybe even Today and Direction. The plots suggest and as time goes on Volume will increase, and that if a market was up on a given week then the percentage return for that week (Today) will tend to be higher. And also correlation table confirms that there is strong relationship between Year and Volume (correlation coefficient of ~0.84).

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.1.3
```

```
library(knitr)
```

```
summary(Weekly)
```

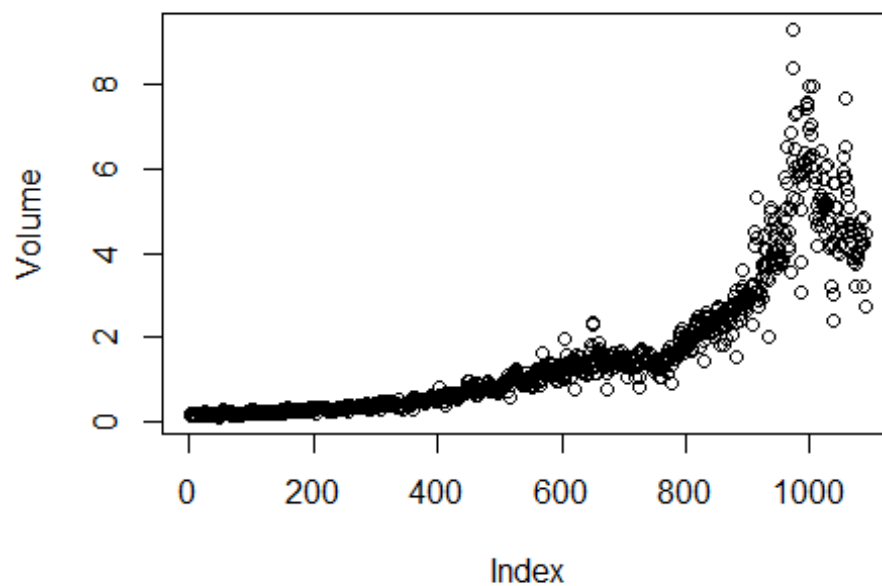
```
##      Year      Lag1      Lag2      Lag3
## Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
## 1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
## Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
## Mean   :2000   Mean    :  0.1506   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      Today
## Min.   :-18.1950   Min.   :-18.1950   Min.    :0.08747   Min.    :-
18.1950
## 1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -
1.1540
## Median :  0.2380   Median :  0.2340   Median :1.00268   Median :
0.2410
## Mean    :  0.1458   Mean    :  0.1399   Mean    :1.57462   Mean    :
0.1499
```

```
## 3rd Qu.: 1.4090 3rd Qu.: 1.4050 3rd Qu.:2.05373 3rd Qu.:
1.4050
## Max. : 12.0260 Max. : 12.0260 Max. :9.32821 Max. :
12.0260
## Direction
## Down:484
## Up :605
##
##
##
##
```

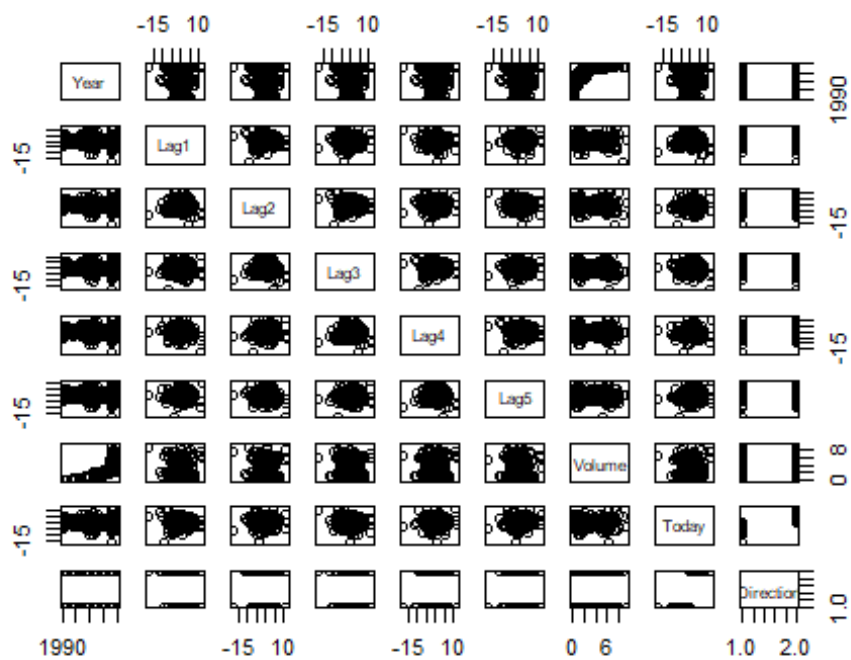
```
cor(Weekly[, -9])
```

```
##          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
```

```
attach(Weekly)
plot(Volume)
```



`pairs(Weekly)`



```
corr_table <- kable(cor(Weekly[, -9]))
corr_table
```

| | Year | Lag 1 | Lag 2 | Lag 3 | Lag 4 | Lag 5 | Volume | To day |
|--------|------|-------|-------|-------|-------|-------|--------|--------|
| Year | 1.0 | - | - | - | - | - | 0.8 | - |
| 1 | 00 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 41 | 0.0 |
| 2 | 00 | 32 | 33 | 30 | 31 | 30 | 94 | 32 |
| 3 | 00 | 28 | 39 | 00 | 12 | 51 | 16 | 45 |
| 4 | | 93 | 00 | 65 | 79 | 91 | | 99 |
| Lag 1 | - | 1.0 | - | 0.0 | - | - | - | - |
| 1 | 0.0 | 00 | 0.0 | 58 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2 | 32 | 00 | 74 | 63 | 71 | 08 | 64 | 75 |
| 3 | 28 | 00 | 85 | 57 | 27 | 18 | 95 | 03 |
| 4 | 93 | | 31 | | 39 | 31 | 13 | 18 |
| Lag 2 | - | - | 1.0 | - | 0.0 | - | - | 0.0 |
| 1 | 0.0 | 0.0 | 00 | 0.0 | 58 | 0.0 | 0.0 | 59 |
| 2 | 33 | 74 | 00 | 75 | 38 | 72 | 85 | 16 |
| 3 | 39 | 85 | 00 | 72 | 15 | 49 | 51 | 67 |
| 4 | 00 | 31 | | 09 | | 95 | 31 | |
| Lag 3 | - | 0.0 | - | 1.0 | - | 0.0 | - | - |
| 1 | 0.0 | 58 | 0.0 | 00 | 0.0 | 60 | 0.0 | 0.0 |
| 2 | 30 | 63 | 75 | 00 | 75 | 65 | 69 | 71 |
| 3 | 00 | 57 | 72 | 00 | 39 | 72 | 28 | 24 |
| 4 | 65 | | 09 | | 59 | | 77 | 36 |
| Lag 4 | - | - | 0.0 | - | 1.0 | - | - | - |
| 1 | 0.0 | 0.0 | 58 | 0.0 | 00 | 0.0 | 0.0 | 0.0 |
| 2 | 31 | 71 | 38 | 75 | 00 | 75 | 61 | 07 |
| 3 | 12 | 27 | 15 | 39 | 00 | 67 | 07 | 82 |
| 4 | 79 | 39 | | 59 | | 50 | 46 | 59 |
| Lag 5 | - | - | - | 0.0 | - | 1.0 | - | 0.0 |
| 1 | 0.0 | 0.0 | 0.0 | 60 | 0.0 | 00 | 0.0 | 11 |
| 2 | 30 | 08 | 72 | 65 | 75 | 00 | 58 | 01 |
| 3 | 51 | 18 | 49 | 72 | 67 | 00 | 51 | 27 |
| 4 | 91 | 31 | 95 | | 50 | | 74 | |
| Volume | 0.8 | - | - | - | - | - | 1.0 | - |
| 1 | 41 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 00 | 0.0 |
| 2 | 94 | 64 | 85 | 69 | 61 | 58 | 00 | 33 |
| 3 | 16 | 95 | 51 | 28 | 07 | 51 | 00 | 07 |
| 4 | | 13 | 31 | 77 | 46 | 74 | | 78 |
| To day | - | - | 0.0 | - | - | 0.0 | - | 1.0 |
| 1 | 0.0 | 0.0 | 59 | 0.0 | 0.0 | 11 | 0.0 | 00 |
| 2 | 32 | 75 | 16 | 71 | 07 | 01 | 33 | 00 |

| | | | | | | | | |
|---|----|----|----|----|----|----|----|----|
| a | 45 | 03 | 67 | 24 | 82 | 27 | 07 | 00 |
| y | 99 | 18 | | 36 | 59 | | 78 | |

- (b) Use the full data set to perform a logistic regression with Direction as the response and the five lag variables plus Volume as predictors. Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so, which ones?

Yes, Lag2 is the statistically significant predictor since its p-value is 0.0296 which is less than 5%.

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

##
## 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
```

- (c) Compute the confusion matrix and overall fraction of correct predictions. Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression.

correct predictions rate is $= (54+557)/1089 = 0.561065 = 56.1065\%$ and so the training error rate is $1 - 0.561065 = 0.438935$ or 43.8935% so the model is very accurate and when the market goes up the model has the accuracy of $(557/(48+557)) =$

0.9206612 = 92.06612 %

the percentage of correct predictions on the training data is $(54+557)/1089$ and when the market goes down the model's accuracy is 11.1570248% of the time $(54/(54+430)) = 0.111570248 = 11.1570248 \%$

```
confm <- predict(fit.glm, type = "response")
pred.glm <- rep("Down", length(confm))
pred.glm[confm > 0.5] <- "Up"
table(pred.glm, Direction)
```

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

- (d) Now fit the logistic regression model using a training data period from 1990 to 2009 with Lag2 as the only predictor. Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is the data from 2010).

We can see that the percentage of correct predictions on the test data is $(3+32)/52 = 0.6730769 = 67.30769 \%$ and so $1-0.6730769 = 0.3269231 = 32.69231\%$ is the test error rate. Note that when the market is up the model's accuracy is $(32/(32+0)) = 100 \%$ which is impressive! and when the market goes down the model's accuracy drops down to $3/(3+17) = 0.15 = 15\%$ So this model is only accurate when the market is up.

```
train <- Weekly[Weekly$Year < 2010, ]
test <- Weekly[!(Weekly$Year < 2010), ]

glm.fit.train <- glm(Direction ~ Lag2, data = train, family = binomial)
glm.probs.test <- predict(glm.fit.train, newdata = test, type =
"response")
glm.pred.test <- ifelse(glm.probs.test > 0.5, "Up", "Down")

table(glm.pred.test, test$Direction)
```

```
##
## glm.pred.test Down Up
##           Down    3  0
##           Up    17 32
```

- (e) Repeat (d) using LDA. We can see that LDA model's accuracy rate is $(7+57)/104 = 0.6153846154$

```
library(MASS)
train_10 <- (Year < 2010)
Weekly.test <- Weekly[!train_10,]
train_10 <- (Year < 2010)
lda_md10 <- lda(Direction ~ Lag2, data = Weekly, subset = train_10)
lda_10_prob <- predict(lda_md10, Weekly.test)
```

```
lda_10_class <- lda_10_prob$class
table(lda_10_class, Weekly.test$Direction)
```

```
##
## lda_10_class Down Up
##      Down      3   0
##      Up       17  32
```

- (f) Repeat (d) using KNN with $K = 1$. KNN model's accuracy rate when $k=1$ is $(8+17)/52 = 0.4807692308$

```
library(class)
train_10 <- (Year < 2010)
train.x_10 <- matrix(Lag2[train_10])
test.x_10 <- matrix(Lag2[!train_10])
train.direction_10 <- Direction[train_10]
Direction_10 = Direction[!train_10]
set.seed(1)
knn_pred_10 <- knn(train.x_10, test.x_10, train.direction_10, k=1)
table(knn_pred_10, Direction_10)
```

```
##      Direction_10
## knn_pred_10 Down Up
##      Down      8  15
##      Up       12  17
```

- (h) Which of these methods appears to provide the best results on this data? The logistic model is the most accurate with an accuracy rate of 67.30769 %
- (i) Experiment with different combinations of predictors, including possible transformations and interactions, for each of the methods. Report the variables, method, and associated confusion matrix that appears to provide the best results on the held out data. Note that you can experiment with values for K in the KNN classifier. After trying different transformations for the different methods, the original Logistic regression and KNN with $k=100$ models had the best accuracy rates.

```
## Load the data
library(ISLR)
```

```
## take a Look
head(Weekly)
```

```
##   Year  Lag1  Lag2  Lag3  Lag4  Lag5  Volume  Today  Direction
## 1 1990  0.816  1.572 -3.936 -0.229 -3.484 0.1549760 -0.270      Down
## 2 1990 -0.270  0.816  1.572 -3.936 -0.229 0.1485740 -2.576      Down
## 3 1990 -2.576 -0.270  0.816  1.572 -3.936 0.1598375  3.514      Up
## 4 1990  3.514 -2.576 -0.270  0.816  1.572 0.1616300  0.712      Up
## 5 1990  0.712  3.514 -2.576 -0.270  0.816 0.1537280  1.178      Up
## 6 1990  1.178  0.712  3.514 -2.576 -0.270 0.1544440 -1.372      Down
```

```

#Logistic Regression with Interaction Lag2:Lag4
Weekly.fit<-glm(Direction~Lag2:Lag4+Lag2, data=Weekly,family=binomial,
subset=train_10)
logWeekly.prob= predict(Weekly.fit, test, type = "response")
logWeekly.pred = rep("Down", length(logWeekly.prob))
logWeekly.pred[logWeekly.prob > 0.5] = "Up"
Direction_10 = Direction[!train_10]
table(logWeekly.pred, Direction_10)

##              Direction_10
## logWeekly.pred Down Up
##              Down    2  0
##              Up    18 32

mean(logWeekly.pred == Direction_10)

## [1] 0.6538462

#KNN with K=10
train.x_10 <- matrix(Lag2[train_10])
test.x_10 <- matrix(Lag2[!train_10])
train.direction_10 <- Direction[train_10]
Direction_10 = Direction[!train_10]
set.seed(1)
knn_pred_10 <- knn(train.x_10, test.x_10, train.direction_10, k=10)
table(knn_pred_10, Direction_10)

##              Direction_10
## knn_pred_10 Down Up
##              Down    9 12
##              Up    11 20

mean(knn_pred_10 == Direction_10)

## [1] 0.5576923

#KNN with k=100
train.x_10 <- matrix(Lag2[train_10])
test.x_10 <- matrix(Lag2[!train_10])
train.direction_10 <- Direction[train_10]
Direction_10 = Direction[!train_10]
set.seed(1)
knn_pred_10 <- knn(train.x_10, test.x_10, train.direction_10, k=100)
table(knn_pred_10, Direction_10)

##              Direction_10
## knn_pred_10 Down Up
##              Down    5  2
##              Up    15 30

```



```
mean(knn_pred_10 == Direction_10)

## [1] 0.6730769
```

4. Using the Boston data set, fit classification models in order to predict whether a given suburb has a crime rate above or below the median. Explore logistic regression, LDA, and KNN models using various subsets of the predictors. Describe your findings.

pair-wise correlation plots shows that variables nox, age, dis, medv may have some sort of correlation with our crime rate. Therefore these may make good predictors for our crim_lvl. So we will run a logistic regression model to identify which of our predictors are significant and then evaluate how effective our model is at predicting if the crime rate will be above or below the median. In the final logistic model nox and medv were the final predictors since the other predictors were omitted due to their high p value which indicated insignificance for the model. The final Logistic model states that as the nox variable increases (nitrogen oxides concentration) our predicted probability of above-median-crime-rate for that neighborhood increases too. The model also states that as the median value of owner-occupied homes increases then the predicted probability for a high crime-rate also increases. Error rate of final Logistic model with a threshold of 50% : 19.2%.

Next step: LDA analysis In the LDA model, the nox, age, and medv variables have a positive correlation with the probability of a high crime rate (above median) while the dis variable has a negative correlation with the predicted probability. In the LDA model, the error rate is a bit lower than the Error rate of the logistic model with the rate at 18.5%.((14 + 13)/146)

Final step: KNN model For the KNN model with k=3 we can see that the error rate is $(4+5)/146 = 0.06164384$ for the KNN with k=5, the error rate is $(5 + 6)/146 = 0.07534247$

So KNN is better than both the Logistic Regression and LDA models in this criteria. From the two KNNs, the best model is the KNN model with a k of 3. Although KNN doesn't tell us which predictors are important, it tells us if we want to lower the crime rate, we should look at what low-crime communities are doing and based on our variables, we should look at that low-crime communities are doing in terms of air pollution, tax rates/tax structures and school funding.

```
```r
load the data
library(MASS)

attach(Boston)
median_crime = median(crim)
#We will create crim_lvl variable that takes on two values: "0" or "1".
#"0" if crime rate below median and "1" if above median.
crim_lvl <- rep(0, 506)
crim_lvl[crim > median_crime] = 1
crim_lvl <- as.factor(crim_lvl)
```

```

Boston_2 <- data.frame(Boston, crim_lvl)
detach(Boston)

pairs(Boston_2)
```

<!-- -->

```r
set.seed(1)
train_13 <- rbinom(506, 1, 0.7)
Boston_2 <- cbind(Boston_2, train_13)
Boston.train <- Boston_2[train_13 == 1,]
Boston.test <- Boston_2[train_13 == 0,]

#fitting a logistic model.
attach(Boston.train)
```

```
The following objects are masked _by_ .GlobalEnv:
##
crim_lvl, train_13
##
```

```r
log_reg <- glm(crim_lvl~nox + age + dis + medv, data = Boston.train, family =
binomial)
summary(log_reg)
```

```
##
Call:
glm(formula = crim_lvl ~ nox + age + dis + medv, family = binomial,
data = Boston.train)
##
Deviance Residuals:
Min 1Q Median 3Q Max
-2.17404 -0.35742 0.00278 0.26418 2.53635
##
Coefficients:
Estimate Std. Error z value Pr(>|z|)
(Intercept) -24.900404 4.019591 -6.195 5.84e-10 ***
nox 38.426015 5.859800 6.558 5.47e-11 ***
age 0.016253 0.009966 1.631 0.1029
dis 0.309361 0.164026 1.886 0.0593 .
medv 0.087237 0.028230 3.090 0.0020 **

```

```

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
(Dispersion parameter for binomial family taken to be 1)
##
Null deviance: 499.02 on 359 degrees of freedom
Residual deviance: 210.55 on 355 degrees of freedom
AIC: 220.55
##
Number of Fisher Scoring iterations: 7
```

```r
#From the p values of coefficients of the previous logistic model age is not
a significant predictor, so we will remove it from the model.
log_reg <- glm(crim_lvl~nox + dis + medv, data = Boston.train, family =
binomial)
summary(log_reg)
```

```
##
Call:
glm(formula = crim_lvl ~ nox + dis + medv, family = binomial,
data = Boston.train)
##
Deviance Residuals:
Min 1Q Median 3Q Max
-2.19958 -0.39388 0.00252 0.26563 2.48475
##
Coefficients:
Estimate Std. Error z value Pr(>|z|)
(Intercept) -23.94414 3.89175 -6.153 7.63e-10 ***
nox 39.78032 5.77733 6.886 5.75e-12 ***
dis 0.23393 0.15697 1.490 0.13615
medv 0.07713 0.02678 2.880 0.00398 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
(Dispersion parameter for binomial family taken to be 1)
##
Null deviance: 499.02 on 359 degrees of freedom
Residual deviance: 213.25 on 356 degrees of freedom
AIC: 221.25
##
Number of Fisher Scoring iterations: 7
```

```r
#now dis has become non significant too

```

```

#dis has now become another un-significant predictor after the removal of
age. So we will remove it until only significant predictors are left.
log_reg <- glm(crim_lvl~nox + medv, data = Boston.train, family = binomial)
summary(log_reg)
```

```
##
Call:
glm(formula = crim_lvl ~ nox + medv, family = binomial, data =
Boston.train)
##
Deviance Residuals:
Min 1Q Median 3Q Max
-2.17657 -0.38729 0.00523 0.30375 2.65695
##
Coefficients:
Estimate Std. Error z value Pr(>|z|)
(Intercept) -19.74864 2.42833 -8.133 4.2e-16 ***
nox 33.97633 3.88025 8.756 < 2e-16 ***
medv 0.06605 0.02524 2.617 0.00887 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
(Dispersion parameter for binomial family taken to be 1)
##
Null deviance: 499.02 on 359 degrees of freedom
Residual deviance: 215.41 on 357 degrees of freedom
AIC: 221.41
##
Number of Fisher Scoring iterations: 6
```

```r
#Now we have our final set of predictors.
detach(Boston.train)

prob_reg <- predict(log_reg, Boston.test, type = 'response')
pred_reg <- rep(0, 146)
pred_reg[prob_reg > 0.5] = 1

ftable <- matrix(data=table(pred_reg, Boston.test$crim_lvl), nrow=2, ncol=2,
 dimnames=list(c("Below median", "Above median"), c("Below",
"Above")))
names(dimnames(ftable)) <- c("predicted", "observed")
print(ftable)
```

```

```

```
observed
predicted Below Above
Below median 63 16
Above median 12 55
```

```r
#LDA analysis
lda_md1 <- lda(crim_lvl~nox + age+ dis+medv, data = Boston_2, subset=
(train_13==1))
lda_md1
```

```
Call:
lda(crim_lvl ~ nox + age + dis + medv, data = Boston_2, subset = (train_13
==
1))
##
Prior probabilities of groups:
0 1
0.4944444 0.5055556
##
Group means:
nox age dis medv
0 0.4690051 51.33371 5.241056 24.95618
1 0.6401758 86.26044 2.498684 20.19670
##
Coefficients of linear discriminants:
LD1
nox 9.32383649
age 0.01342745
dis -0.08315746
medv 0.01499271
```

```r
lda_preds <- predict(lda_md1, Boston.test)
lda_class <- lda_preds$class

dat <- matrix(data=table(lda_class, Boston.test$crim_lvl), nrow=2, ncol=2,
 dimnames=list(c("Below median", "Above median"), c("Below",
"Above")))
names(dimnames(dat)) <- c("predicted", "observed")
print(dat)
```

```

```

```

observed
predicted Below Above
Below median 62 14
Above median 13 57
```

```r
#KNN model
#k=3
train.x_13 <- cbind(Boston.train$nox, Boston.train$tax, Boston.train$pratio)
test.x_13 <- cbind(Boston.test$nox, Boston.test$tax, Boston.test$pratio)
set.seed(1)
knn_pred_13 <- knn(train.x_13, test.x_13, Boston.train$crim_lvl, k=3)
table(knn_pred_13, Boston.test$crim_lvl)
```

```
##
knn_pred_13 0 1
0 71 5
1 4 66
```

```r
#KNN with k=5

knn_pred_13_2 <- knn(train.x_13, test.x_13, Boston.train$crim_lvl, k=5)
table(knn_pred_13_2, Boston.test$crim_lvl)
```

```
##
knn_pred_13_2 0 1
0 69 5
1 6 66
```

```