# STATS 415 HW 6

Jiahao Cheng

March 2025

## Problem 1

(a) As the summary shows(Figure 1), both intercept and Lag2 are significant(p-value $< 0.05$), but Lag1 is not significant(p-value $> 0.05$).

```
> ## (a)
> model_a <- glm(Direction ~ Lag1 + Lag2, data = Weekly, family = binomial)
> summary(model_a)

Call:
glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = Weekly)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.22122    0.06147   3.599 0.000319 ***
Lag1        -0.03872    0.02622  -1.477 0.139672
Lag2         0.06025    0.02655   2.270 0.023232 *
---
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: 1488.2  on 1086  degrees of freedom
AIC: 1494.2

Number of Fisher Scoring iterations: 4
```

Figure 1: Logistic Regression on Lag1 and Lag2

(b) As the summary shows(Figure 2), both intercept and Lag2 are significant(p-value $< 0.05$), but Lag1 is not significant(p-value $> 0.05$).

```
> ## (b)
> model_b <- glm(Direction ~ Lag1 + Lag2, data = Weekly[-1, ], family = binomial)
> summary(model_b)

Call:
glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = Weekly[-1,
    ])

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.22324    0.06150   3.630 0.000283 ***
Lag1        -0.03843    0.02622  -1.466 0.142683
Lag2         0.06085    0.02656   2.291 0.021971 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1494.6  on 1087  degrees of freedom
Residual deviance: 1486.5  on 1085  degrees of freedom
AIC: 1492.5

Number of Fisher Scoring iterations: 4
```

Figure 2: Logistic Regression on Lag1 and Lag2 without the first observation

(c) From Figure 3, the probability that the first observation's direction is Up is $0.571 > 0.5$. So we predict the direction is Up, but the actual direction is Down, so the prediction is incorrect.

```
> ## (c)
> first <- Weekly[1, ]
> prob <- predict(model_b, newdata = first, type = "response")
> prob
        1
0.5713923
```

Figure 3: Prediction Result

(d) Figure 4 shows the code.

```
> ## (d)
> n <- nrow(Weekly)
> errors <- rep(NA, n)
> for (i in 1:n) {
+
+   train_data <- Weekly[-i, ]
+
+   model <- glm(Direction ~ Lag1 + Lag2, data = train_data, family = binomial)
+
+   prob <- predict(model, newdata = Weekly[i, ], type = "response")
+
+   pred <- ifelse(prob > 0.5, "Up", "Down")
+
+   actual <- as.character(Smarket$Direction[i])
+   errors[i] <- ifelse(pred != actual, 1, 0)
+ }
```

Figure 4: LOOCV

(e) From Figure 5, we can see that the LOOCV error rate is 0.503. It indicates that the logistic regression's accuracy is around 0.497, which is not high. This implies that the model might not generalize well, and using Lag1 and Lag2 to make prediction is not sufficient.

```
> ## (e)
> error_rate <- mean(errors)
> print(paste("LOOCV Error Rate:", round(error_rate, 3)))
[1] "LOOCV Error Rate: 0.503"
```

Figure 5: LOOCV error rate

# Problem 2

(a) Figure 6 shows the regression result. All three polynomial terms and the intercept are significant(p-value $< 0.001$). The cubic model seems to capture non-linear trend.
Figure 7 shows the plot, a smooth curve fitting the scatter of points. But perform a bit bad around $dis = 10 - 12$

```
> model_poly3 <- lm(nox ~ poly(dis, 3), data = Boston)
> summary(model_poly3)

Call:
lm(formula = nox ~ poly(dis, 3), data = Boston)

Residuals:
      Min        1Q    Median        3Q       Max
-0.121130 -0.040619 -0.009738  0.023385  0.194904

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.554695   0.002759 201.021  < 2e-16 ***
poly(dis, 3)1 -2.003096   0.062071 -32.271  < 2e-16 ***
poly(dis, 3)2  0.856330   0.062071  13.796  < 2e-16 ***
poly(dis, 3)3 -0.318049   0.062071  -5.124 4.27e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.06207 on 502 degrees of freedom
Multiple R-squared:  0.7148,    Adjusted R-squared:  0.7131
F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16

> plot(Boston$dis, Boston$nox, main = "Cubic Polynomial: nox vs dis",
+      xlab = "dis", ylab = "nox", pch = 18, col = "red")
> dis_values <- seq(min(Boston$dis), max(Boston$dis), length.out = 100)
> predicted_nox <- predict(model_poly3, newdata = data.frame(dis = dis_values))
> lines(dis_values, predicted_nox, col = "blue", lwd = 2)
```

Figure 6: polynomial 3

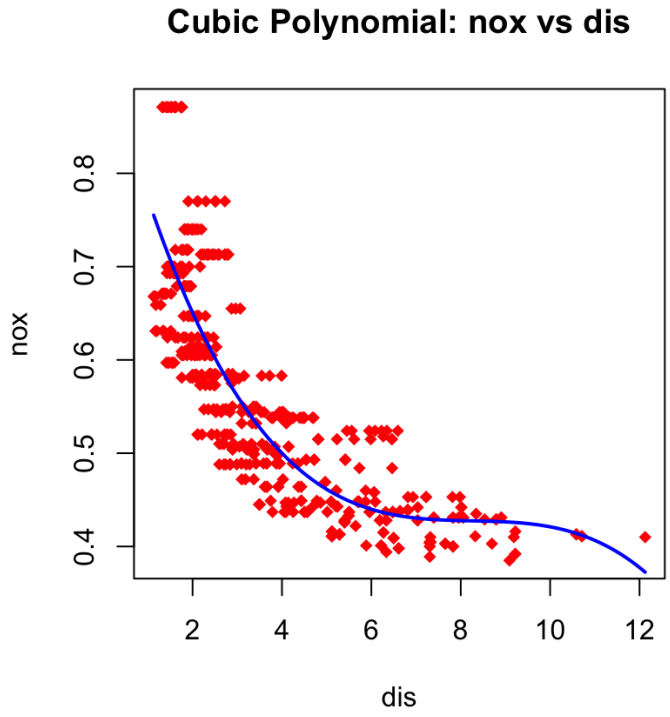


Figure 7: plot of polynomial 3

(b) From Figures 8, 9, 10, we can see that as the polynomial degree increases, the RSS is decreasing, the model fits the data better, but has higher risk of overfitting. For example, degree 10 plot shows a wiggle at the end, that is weird.

```
> ## (b)
> x <- Boston$dis
> y <- Boston$nox
> rss <- numeric(10)
> for (d in 1:10) {
+   model <- lm(nox ~ poly(dis, d), data = Boston)
+   rss[d] <- sum(residuals(model)^2)
+ }
> plot(1:10, rss, type = "b", pch = 19,
+      xlab = "Polynomial Degree", ylab = "RSS",
+      main = "RSS vs Polynomial Degree")
> plot(x, y, main = "Polynomial Degrees 1 to 10",
+      xlab = "dis", ylab = "nox", col = "grey", pch = 20)
> dis_grid <- seq(min(x), max(x), length.out = 300)
> colors <- rainbow(10)
> for (d in 1:10) {
+   model <- lm(nox ~ poly(dis, d), data = Boston)
+   pred <- predict(model, newdata = data.frame(dis = dis_grid))
+   lines(dis_grid, pred, col = colors[d], lwd = 1.5)
+ }
> legend("topright", legend = paste("Degree", 1:10), col = colors, lwd = 5, cex = 0.8)
```

Figure 8: Plot code

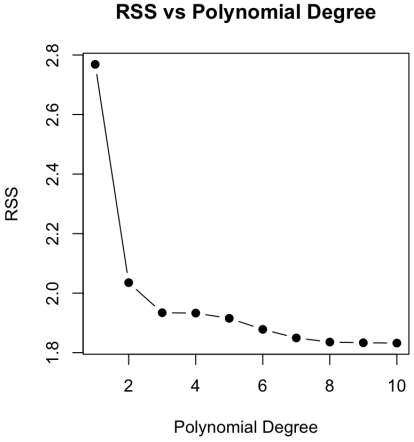


Figure 9: RSS vs Polynomial Degree

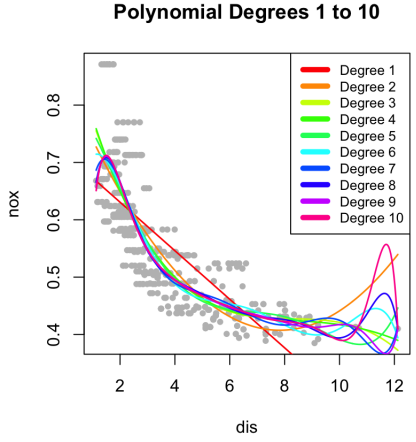


Figure 10: Polynomial Degree 1 to 10

(c) Figure 11 shows that, the optimal degree is 3, which has the lowest error rate - 0.0039.

```
> ## (c)
> library(boot)
> cv_error <- numeric(10)
> for (d in 1:10) {
+   model <- glm(nox ~ poly(dis, d), data = Boston)
+   cv_result <- cv.glm(Boston, model, K = 10)
+   cv_error[d] <- cv_result$delta[1]
+   print(cv_error[d])
+ }
[1] 0.005513459
[1] 0.004083877
[1] 0.003873764
[1] 0.003912737
[1] 0.004129041
[1] 0.00544039
[1] 0.01029739
[1] 0.01365189
[1] 0.0148662
[1] 0.008931853
> optimal_degree <- which.min(cv_error)
> optimal_degree
[1] 3
```

Figure 11: 10-Folds Cross Validation

(d) Figure 12 shows the code. According to the summary, all spline basis terms are significant(p-value $< 0.01$). The relationship between dis and nox is non-linear and well captured by the spline.
The degree of freedom is decided by the number of knots. In the case of cubic splines, $df = n_{knots} + 3$. When the df is specified, the knots are automatically placed at uniform percentiles of the data. Therefore, the number of knots in this model is 1.
Figure 13 shows the fitting result.

```
> ## (d)
> library(splines)
> model_spline <- lm(nox ~ bs(dis, df = 4), data = Boston)
> # Summary of the model
> summary(model_spline)

Call:
lm(formula = nox ~ bs(dis, df = 4), data = Boston)

Residuals:
      Min        1Q    Median        3Q       Max
-0.124622 -0.039259 -0.008514  0.020850  0.193891

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.73447    0.01460  50.306  < 2e-16 ***
bs(dis, df = 4)1 -0.05810    0.02186  -2.658  0.00812 **
bs(dis, df = 4)2 -0.46356    0.02366 -19.596  < 2e-16 ***
bs(dis, df = 4)3 -0.19979    0.04311  -4.634 4.58e-06 ***
bs(dis, df = 4)4 -0.38881    0.04551  -8.544  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.06195 on 501 degrees of freedom
Multiple R-squared:  0.7164,    Adjusted R-squared:  0.7142
F-statistic: 316.5 on 4 and 501 DF,  p-value: < 2.2e-16

> plot(Boston$dis, Boston$nox, col = "gray", pch = 20,
+      main = "Regression Spline Fit (df = 4)", xlab = "dis", ylab = "nox")
> dis_grid <- seq(min(Boston$dis), max(Boston$dis), length.out = 300)
> pred_spline <- predict(model_spline, newdata = data.frame(dis = dis_grid))
> lines(dis_grid, pred_spline, col = "blue", lwd = 2)
```

Figure 12: Cubic Spline Code

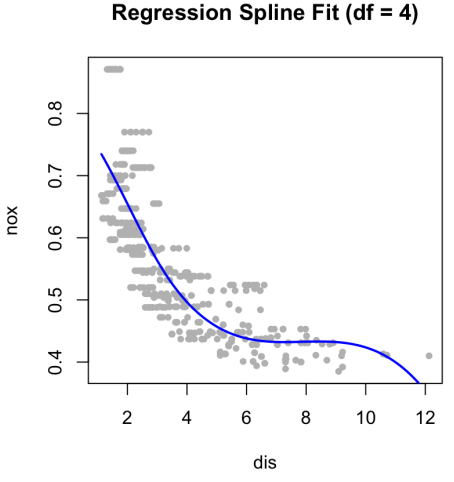


Figure 13: Cubic Spline Plot

(e) From Figures 14, 15, 16, we can see that as the degree of freedom increases, the RSS is generally decreasing(except df 8 to 9), the model fits the train data better, but has higher risk of overfitting. For example, degree 10 plot shows a wiggle curve around $dis = 2$, that is weird.

```
> ## (e)
> x <- Boston$dis
> y <- Boston$nox
> rss_values <- numeric(8)
> dfs <- 3:10
> plot(x, y, col = "gray", pch = 20,
+      main = "Regression Spline Fits (df = 3 to 10)",
+      xlab = "dis", ylab = "nox")
> x_grid <- seq(min(x), max(x), length.out = 300)
> colors <- rainbow(length(dfs))
> for (i in seq_along(dfs)) {
+   df_val <- dfs[i]
+   model <- lm(nox ~ bs(dis, df = df_val), data = Boston)
+
+   rss_values[i] <- sum(residuals(model)^2)
+
+   y_pred <- predict(model, newdata = data.frame(dis = x_grid))
+   lines(x_grid, y_pred, col = colors[i], lwd = 1.5)
+ }
> legend("topright", legend = paste("df =", dfs), col = colors, lwd = 3, cex = 0.8)
> plot(3:10, rss_values, type = "b", pch = 19,
+      xlab = "Spline Df", ylab = "RSS",
+      main = "RSS vs Spline Df")
```

Figure 14: Plot code

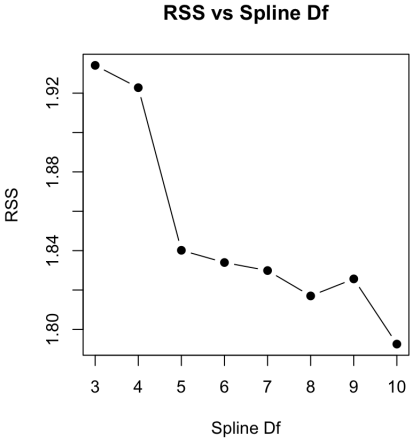


Figure 15: RSS vs Degree of Freedom

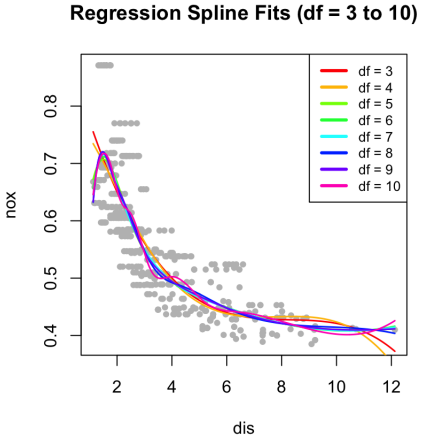


Figure 16: Df 3 to 10

(f) Figure 17 shows the code of 10 folds cross validation. Degree 7 has the smallest

error, 0.003709, so I choose degree freedom 7 as the best degree.

```
> ## (f)
> library(boot)
> set.seed(123)
> cv.error_df <- rep(3, 10)
> for(i in 3:10) {
+   df_val <- i
+   fit <- glm(nox ~ bs(dis, df = df_val), data = Boston)
+   cv.error_df[i] <- cv.glm(Boston, fit, K = 10)$delta[1]
+   print(cv.error_df[i])
+ }
[1] 0.003879744
[1] 0.003880244
[1] 0.00370928
[1] 0.003710156
[1] 0.003709786
[1] 0.003719635
[1] 0.003715567
[1] 0.003711569
There were 26 warnings (use warnings() to see them)
> which.min(cv.error_df)
[1] 5
```

Figure 17: Logistic test error