Linear regression

Chapter 3 of ISLR/ISLP mainly concerns linear regression. It is worth reading, but students should already be familiar with most of the things it discusses, so I will only talk about a few of them.

Consider the linear regression model

$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j X_{ji} + u_i, \tag{1}$$

where p may be quite large.

Suppose we do not know which of the p regressors to include. Since each of them may or may not belong, there are 2^p models to consider.

- When $p = 20$, $2^p = 1,048,576 \approx 10^6$.
- When $p = 30$, $2^p = 1,073,741,824 \approx 10^9$.
- When $p = 40$, $2^p = 1,099,511,627,776 \approx 10^{12}$.

We cannot estimate all possible regression models! But we can follow a **selection rule** and estimate a number of them.

Forward selection starts by estimating p models with a constant and one regressor each.

- Keep whichever added regressor gives the best fit.
- Now add each of the $p-1$ remaining regressors to this model. Keep the newly-added regressor that gives the best fit.
- Now add each of the $p-2$ remaining regressors. Keep the newly-added regressor that gives the best fit.
- Continue in this way until the fit does not improve very much when we add an additional regressor.

Deciding when the fit does not improve very much can be done in various ways.

We could use the t-statistic on the added regressor, or the **AIC** (**Akaike Information Criterion**) or the **BIC** (**Bayesian Information Criterion**).

Backward selection starts by estimating a model that includes all p regressors. It does not work if $p \geq n - 1$.

- Remove whichever regressor has the highest P value (or the smallest t-statistic).
- Re-estimate the model with only $p - 1$ regressors. Once again, remove whichever regressor has the highest P value.
- Stop dropping regressors when we have to remove one with a small P value, or when a criterion like AIC or BIC starts to move in the wrong direction.

A third approach is **mixed selection**. It works like forward selection, but included regressors are dropped when their P value gets too high.

Forward and mixed selection can handle cases with $p > n$, but the final model must have $p < n - 1$ if it is not to fit perfectly within sample.

Let d be the chosen number of regressors. As $d \to n - 1$, the MSE goes to 0, but the prediction errors for the test data eventually become large. Thus d acts like a tuning parameter.

The usual definition of AIC is for models estimated by maximum likelihood:

$$\mathrm{AIC} = 2\log L(\boldsymbol{\theta} \mid \boldsymbol{y}) - 2d, \tag{2}$$

where d is the number of parameters estimated.

In the linear regression case,

$$\log L = \frac{-n}{2}\log(\mathrm{SSR}) + \text{constant}. \tag{3}$$

Thus

$$\mathrm{AIC} = -n\log(\mathrm{SSR}) - 2d. \tag{4}$$

We want to maximize AIC when it is written in this form.

Alternatively, we can minimize

$$\mathrm{AIC}' = \frac{n}{2}\log(\mathrm{SSR}) + d. \tag{5}$$

The penalty for AIC does not increase with n. This means that it may choose a model with too many parameters asymptotically.

The most popular criterion function that will not do this is the **Bayesian information criterion**, or **BIC**.

It is sometimes called the **Schwarz information criterion**.

For models estimated by maximum likelihood, BIC is

$$\mathrm{BIC} = 2\log L(\boldsymbol{\theta}\,|\,\boldsymbol{y}) - d\log(n). \tag{6}$$

This is similar to (2), and once again we maximize it.

- For $n > 7$, $\log(n) > 2$. Thus the BIC penalty is greater than the AIC penalty for almost all sample sizes.
- Since it seems reasonable to penalize flexibility more heavily as n increases, BIC is perhaps more attractive than AIC.
- However, it can lead to models with a lot fewer parameters than AIC, perhaps fewer than optimal.

Nonlinearities in Linear Regression

Using linear regression does not mean that $\mathrm{E}(y)$ must be a linear function of the regressors.

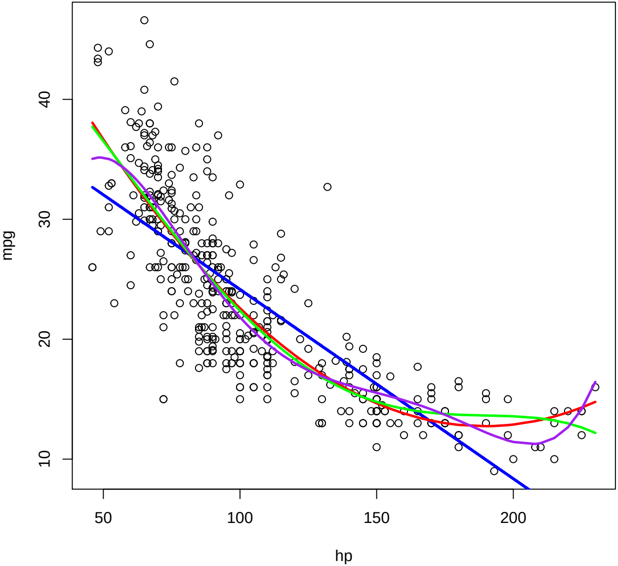
We can include all sorts of nonlinear functions of the original regressors instead of, or in addition, to them.

Examples include x^2, x^3, $\log x$, $\exp x$, $\sqrt{x}$, x_1x_2, and $\sqrt{x_1x_2}$. Using polynomials is popular but risky.

Consider the regression of `MPG` (miles per gallon) on `HP` (horsepower) in Figure 3.8 of ISLR/ISLP. Note that $n = 392$, but data are not i.i.d.

- When just x and x^2 are included, they are both highly significant.
- When either x^3 or both x^3 and x^4 are added, the additional ones are not significant.
- But when x^3 through x^5 are added, all regressors are significant, with t-statistics between 2.84 and 3.21.

Figure 3.1 (MPG vs. HP)



In Figure 3.1, the linear fit is in blue, the quadratic in red, the fourth-order in green, and the fifth-order in purple.

The linear fit is pretty poor, especially in the middle and at both ends.

The quadratic fit is a lot better, but it shows `MPG` increasing with `HP` for large values of `HP`.

The fourth-order fit is similar to the quadratic except at the far right; recall that x^3 and x^4 were both insignificant.

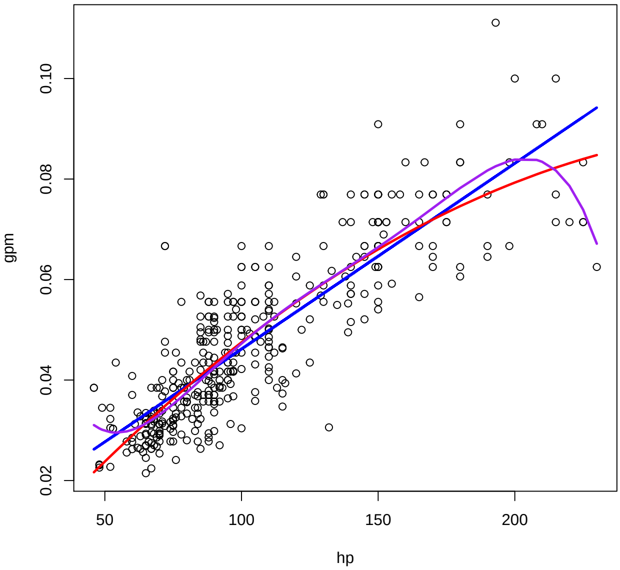
The fifth-order fit is crazy at both ends, especially at the far right.

Imagine what the predicted MPG is for a car with 400 horsepower!

It is always hard to predict y for values of x far from what we have observed. More flexible models (in this case, higher-degree polynomials) make it harder.

Using MPG as the outcome is stupid. It makes more sense to use gallons per mile (GPM). See Figure 3.2.

Figure 3.2 (GPM vs. HP)



The linear and quadratic fits in Figure 3.2 look a lot better than the ones in Figure 3.1.

There is less curvature in the data, so the quadratic curve is quite gentle. It will have $\partial\texttt{MPG}/\partial\texttt{HP}$ positive for sensible values of horsepower (at least by 1970s standards).

The fifth-order polynomial is a disaster, despite *t*-statistics between 2.70 and 2.94. A car with 300 horsepower will use no gas at all!

Both figures display heteroskedasticity. It is less severe in Figure 3.2.

There is not much discussion of this in ISLR/ISLP, and no discussion of correlation within clusters (e.g. by manufacturer).

- Heteroskedasticity may be a sign of misspecification.
- Heteroskedasticity leads to inefficient estimates.
- Heteroskedasticity means that prediction intervals are incorrect.

ISLR/ISLP talks about **prediction intervals** without really saying what they are. More about them later.

Comparing KNN with Linear Regression

We can use KNN with the mileage data.

Instead of using cross-validation to choose K and a test sample to evaluate the fit, I simply tried three values of K: 5, 9, and 13.

I used the `knn.reg()` function in the `FNN` library.

Visually, $K = 13$ seems like the best choice, although it must fit less well than $K = 5$ and $K = 9$.

Residual standard errors are 0.007868 for $K = 5$, 0.007903 for $K = 9$, and 0.008000 for $K = 13$.

This compares with 0.008447 for quadratic regression and 0.008646 for linear regression.

So KNN fits quite a bit better within sample. In terms of **predictive $\boldsymbol{R}^2$**, $K = 7$ (not shown) beats $K = 5$ and $K = 9$.

Figure 3.3 ($K = 5$)

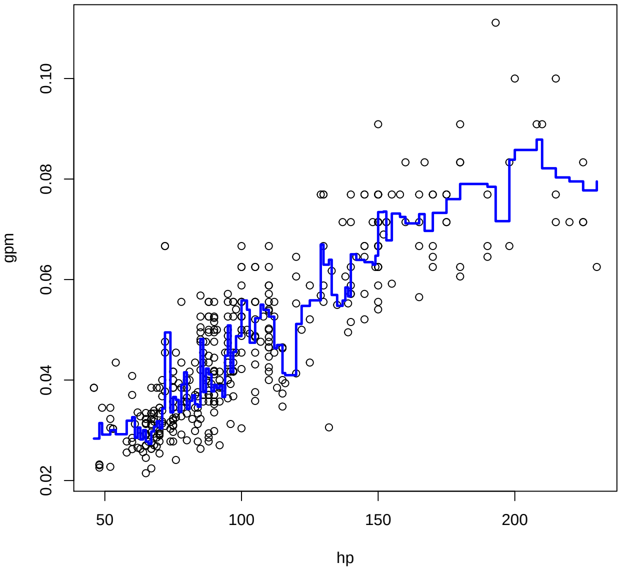


Figure 3.4 ($K = 9$)

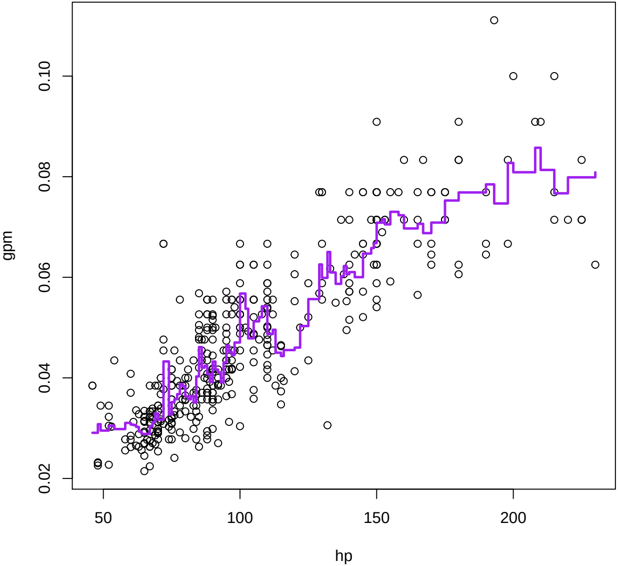


Figure 3.5 ($K = 13$)

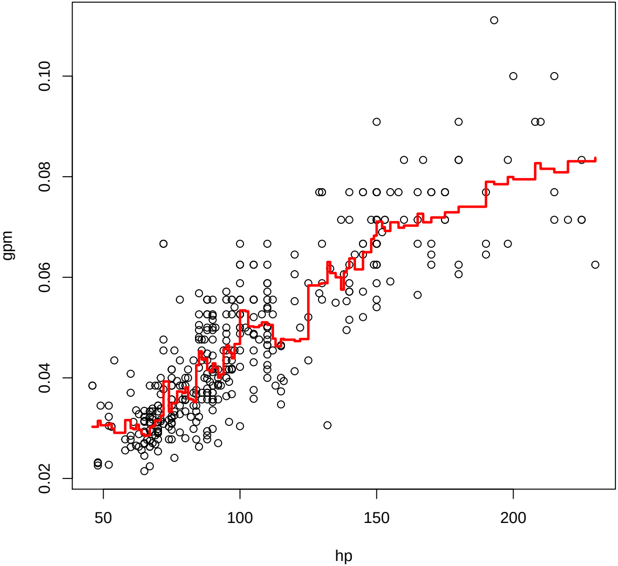
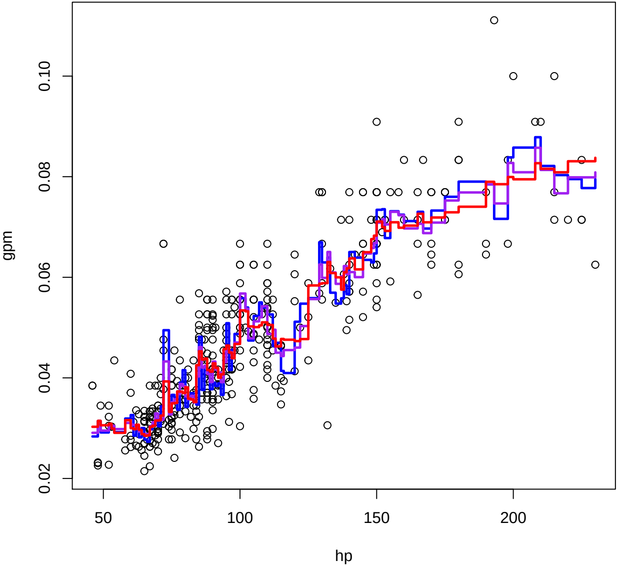


Figure 3.6 (three values of *K*)



Many statistical properties of KNN are shared by other methods that are also completely non-parametric, such as kernel regression and smoothing splines.

For those methods, the fits will look smoother (no staircase effects), but the MSEs will be similar to KNN.

At the end of the chapter, there are some other figures that illustrate how linear regression compares with KNN.

- These are for completely artificial data, with 50 observations.
- In Figures 3.18 to 3.20, the test MSE for KNN regression is shown as a function of $1/K$.
- The test MSE for least squares is a dashed horizontal line.
- The closer to linear the true model is, the better OLS works.
- In Figure 3.19, KNN beats linear regression handily when $f(x)$ is very nonlinear, but just slightly and only for some values of K when $f(x)$ is not very nonlinear.

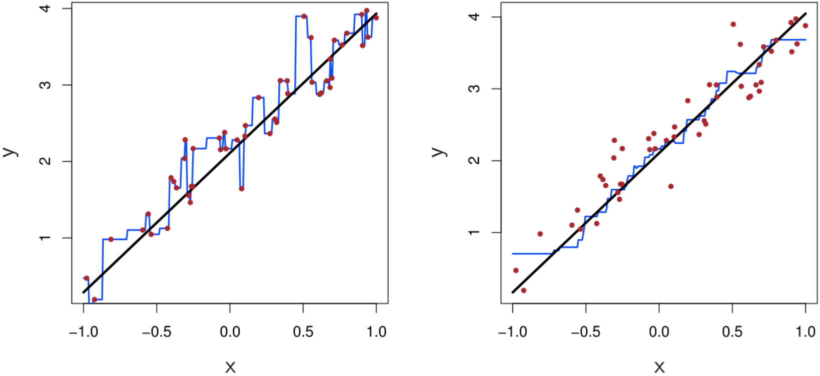


FIGURE 3.17. *Plots of $\hat{f}(X)$ using KNN regression on a one-dimensional data set with 50 observations. The true relationship is given by the black solid line.* Left: *The blue curve corresponds to $K = 1$ and interpolates (i.e. passes directly through) the training data.* Right: *The blue curve corresponds to $K = 9$, and represents a smoother fit.*

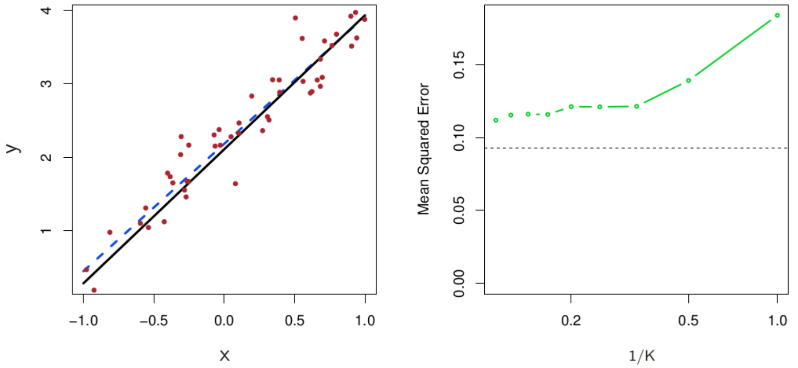


FIGURE 3.18. *The same data set shown in Figure 3.17 is investigated further.* Left: *The blue dashed line is the least squares fit to the data. Since $f(X)$ is in fact linear (displayed as the black line), the least squares regression line provides a very good estimate of $f(X)$.* Right: *The dashed horizontal line represents the least squares test set MSE, while the green solid line corresponds to the MSE for KNN as a function of $1/K$ (on the log scale). Linear regression achieves a lower test MSE than does KNN regression, since $f(X)$ is in fact linear. For KNN regression, the best results occur with a very large value of K, corresponding to a small value of $1/K$.*

Comparing KNN with Linear Regression

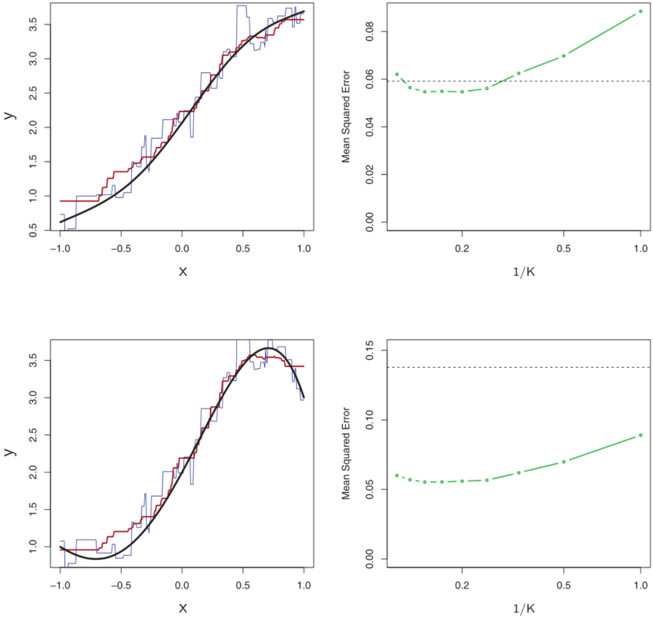


Figure 3.20 illustrates a serious problem with all methods that attempt to be very flexible, namely, the **curse of dimensionality**.

- As p increases, the number of nearby points declines. For a lattice on the unit hypercube to have distance 0.01 between points, we need 101 points for $p = 1$, 101^2 points for $p = 2$, and so on.
- For fixed n, the points get much further apart as p increases. With even modest collinearity, points may be very sparse in some areas.
- If we only have, say, 1000 points, then the average (lattice) distance between them will be $1/999 = 0.0010$ for $p = 1$, $1/33 = 0.0303$ for $p = 2$, and $1/9 = 0.1111$ for $p = 3$.
- This means that the K points nearest to $\boldsymbol{x}_0$ become much further away as p increases, causing bias to increase.
- We can partially compensate by making K smaller as p increases, but that increases variance.
- Whatever we do, test MSE will increase. It also increases for linear regression, but much more slowly.

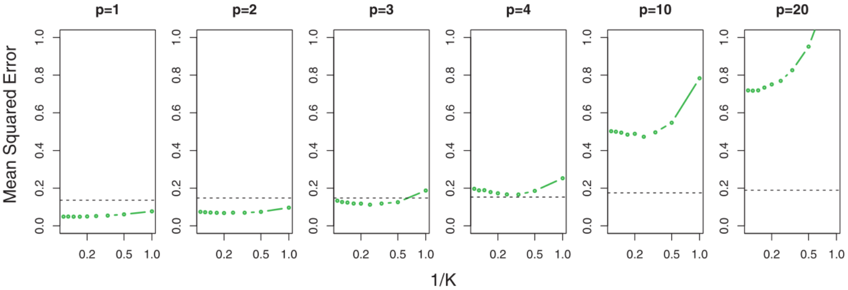


FIGURE 3.20. *Test MSE for linear regression (black dashed lines) and KNN (green curves) as the number of variables p increases. The true function is non–linear in the first variable, as in the lower panel in Figure 3.19, and does not depend on the additional variables. The performance of linear regression deteriorates slowly in the presence of these additional noise variables, whereas KNN's performance degrades much more quickly as p increases.*

Even with a great deal of nonlinearity, linear regression massively outperforms KNN for $p \geq 10$.

Leverage and Influence

For any prediction method, some of the observations in the training sample have more **influence** on the predictions than others.

For KNN, only the K observations nearest $\boldsymbol{x}_0$ have any influence.

For smoothing methods like kernel regression and splines, the influence of $\boldsymbol{x}_i$ usually diminishes as $\boldsymbol{x}_i$ gets further from $\boldsymbol{x}_0$.

But for linear (and polynomial) regression, every observation has some influence, especially ones that are extreme and may be far from $\boldsymbol{x}_0$.

ISLR/ISLP discusses this, but their discussion is limited.

Consider the linear regression model

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{u}, \tag{7}$$

where $\boldsymbol{X}$ is $n \times p + 1$. Dropping the i^{th} observation is equivalent to adding a regressor $\boldsymbol{e}_i$ with i^{th} element 1 and all others 0.

This regression is

$$\boldsymbol{y} = \boldsymbol{X\beta} + \alpha_i \boldsymbol{e}_i + \boldsymbol{u}, \tag{8}$$

The fitted values and residuals from regression (8) are given by

$$\boldsymbol{y} = \boldsymbol{X}\hat{\boldsymbol{\beta}}^{(i)} + \hat{\alpha}\boldsymbol{e}_i + \boldsymbol{M_Z y}, \tag{9}$$

where $\hat{\boldsymbol{\beta}}^{(i)}$ is the vector of OLS estimates based on all observations except the i^{th}, and $\boldsymbol{M_Z}$ projects off $\mathcal{S}(\boldsymbol{X}, \boldsymbol{e}_i)$.

A fair amount of algebra (ETM, Chapter 2) shows that

$$\hat{\boldsymbol{\beta}}^{(i)} - \hat{\boldsymbol{\beta}} = \frac{-1}{1-h_i}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}_i^\top \hat{u}_i. \tag{10}$$

Here h_i is the i^{th} diagonal of the **hat matrix** $\boldsymbol{P_X}$. Thus how influential an observation is depends on both $\hat{u}_i$ and h_i.

The value of h_i determines how much **leverage** observation i has got.

When we care about prediction, it may be more natural to measure influence as the impact of each observation on its own residual (and fitted value):

$$\hat{u}_i^{(i)} - \hat{u}_i = \frac{\hat{u}_i}{1-h_i} - \hat{u}_i = \frac{\hat{u}_i - (1-h_i)\hat{u}_i}{1-h_i} = \frac{h_i}{1-h_i}\hat{u}_i. \tag{11}$$

Influential observations either have large residuals (they are **outliers**), or high leverage, or both.

- Observations with large h_i are said to be **leverage points**.
- It is not hard to show that $0 \leq h_i \leq 1$ and that $\sum_{i=1}^{n} h_i = p+1$.
- Thus, a high-leverage observation is one for which h_i is large relative to $(p+1)/n$.
- When $\hat{u}_i$ is also large, observation i is bound to be **influential**.

A sample is said to be **balanced** when the h_i do not vary much. It is **perfectly balanced** if they do not vary at all.

Figure 3.7 (h_i against horsepower)

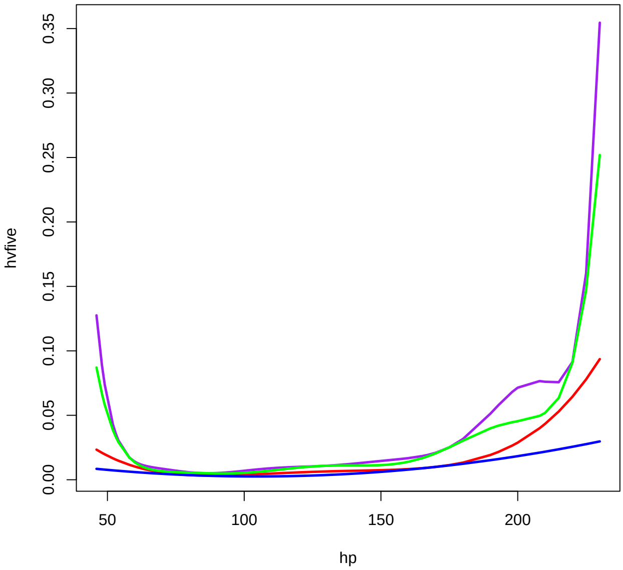


Figure 3.7 graphs h_i against horsepower for four different models of gallons per mile.

The linear model is in blue, the quadratic in red, the fourth-order in green, and the fifth-order in purple.

- The values of h_i are always smallest for "average" values of HP and largest for the largest value.
- For the linear model, the h_i vary moderately.
- For the quadratic model, they vary considerably.
- For the 4^{th} order and 5^{th} order polynomials, they vary extremely. For these models, the largest and smallest values of horsepower have extremely high leverage.

This example illustrates the fact that polynomials often produce extreme and unrealistic predictions for values of $\boldsymbol{x}_0$ near (or beyond) the limits of the observed $\boldsymbol{x}_i$.