STA4026S – Honours Analytics
Section A – Theory and Application of Supervised Learning
Lecture 4 – Non-linear Models

Stefan S. Britz
stefan.britz@uct.ac.za

Department of Statistical Sciences
University of Cape Town

Introduction

- The regression and classification models we considered so far were predicated on the assumption (even after regularisation) that the input variables map **linearly** to the response
- In reality, this will usually be an approximation at best, and wildly over simplistic at worst
- Therefore, although these models offer clear interpretation and inference, this often comes at the cost of predictive power
- We will now go beyond linearity, increasing model flexibility (higher variance) in an attempt to improve fit (lower bias)

Non-linear models

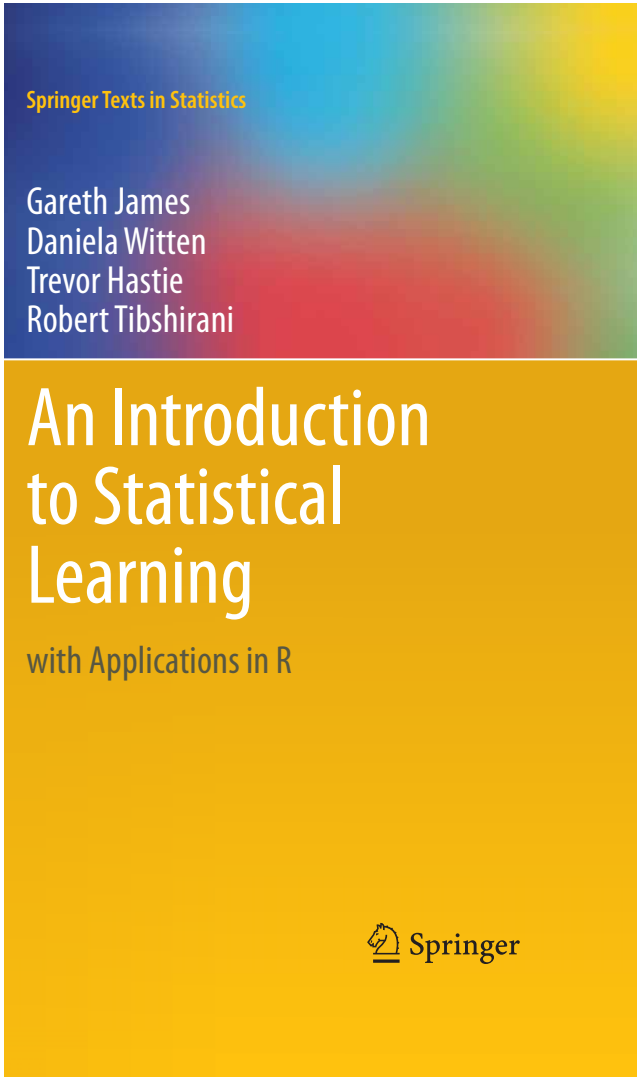
Non-linear models are many and varied, both for regression and classification

Some of the key parametric models include:

- polynomial regression
- step functions
- regression and smoothing splines
- multivariate adaptive regression splines (MARS)
- local regression methods (e.g. loess and lowess)
- generalised additive models (GAMs)

For reasons of scope, we will only cover **polynomial regression** in this lecture, as well as one non-parametric method in K-nearest neighbours (KNN)

Suggested reading



Chapter 7

Polynomial regression

Polynomial regression

- A simple extension of linear regression
- The relationship between the independent variable(s) and the dependent variable is modelled as an d^{th}-degree polynomial
- Allows us to capture more complex, non-linear relationships
- Note that the model must be pre-specified (must decide on formulation beforehand)
- This can be applied in both regression and classification contexts

Polynomial regression – quantitative response

The polynomial regression model for a single predictor variable can be represented as follows:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \cdots + \beta_d X^d + \epsilon. \tag{1}$$

We need to select an appropriate degree

Too high d results in an overly complex model that will overfit on the training data

In practice it is unusual to use $d > 4$, since these highly flexible models can perform especially poorly near the boundaries of the observed predictors

Example 5 – Auto

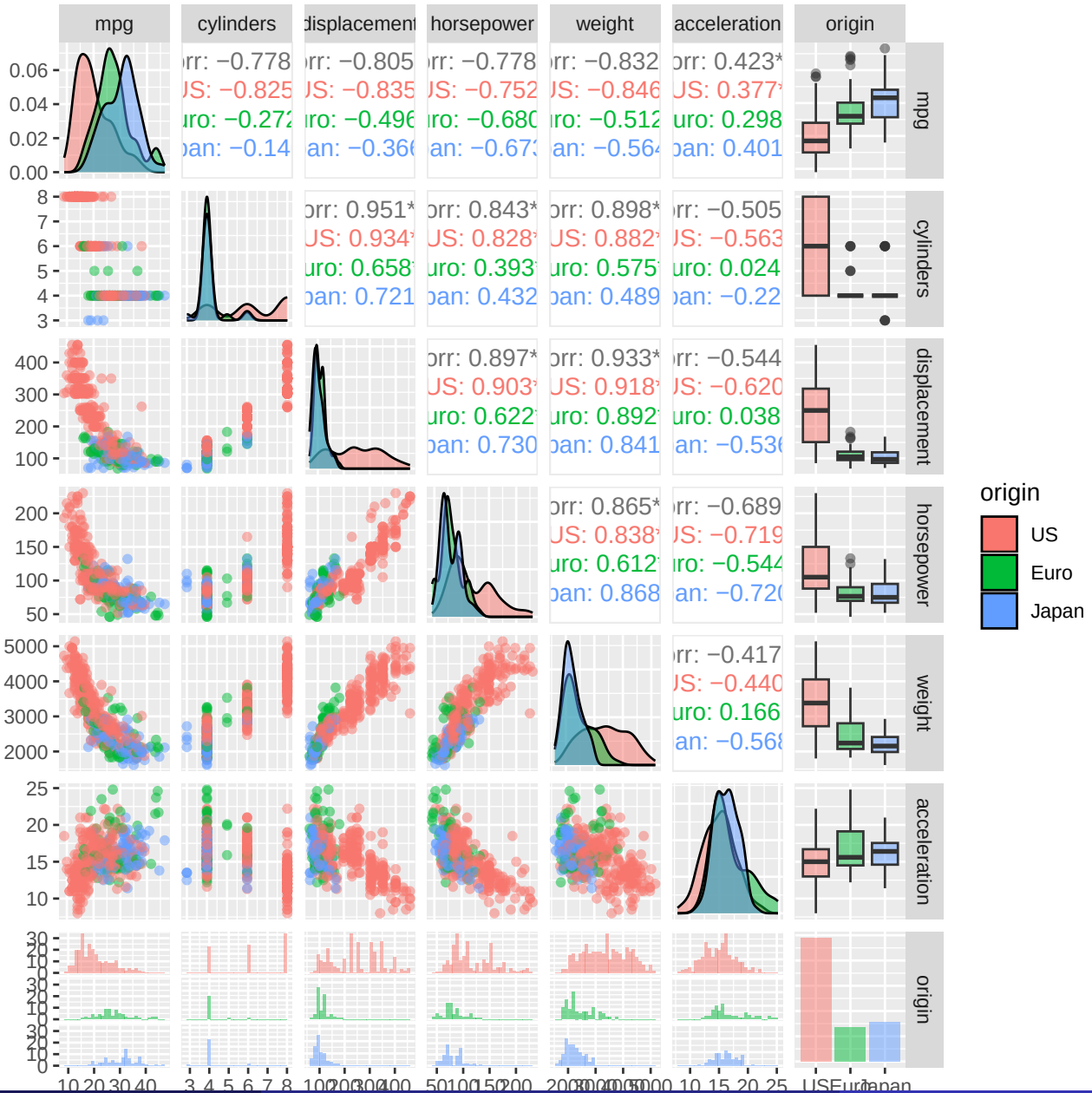
Consider the well-known ‘Auto‘ dataset, available in the ‘ISLR‘ package

Contains measurements on the mileage, engine specifications, and manufacturing information for 392 vehicles

A sensible relationship to model is how a vehicle's mileage depends on its specifications.

First, visually explore the relationships between the numeric variables.

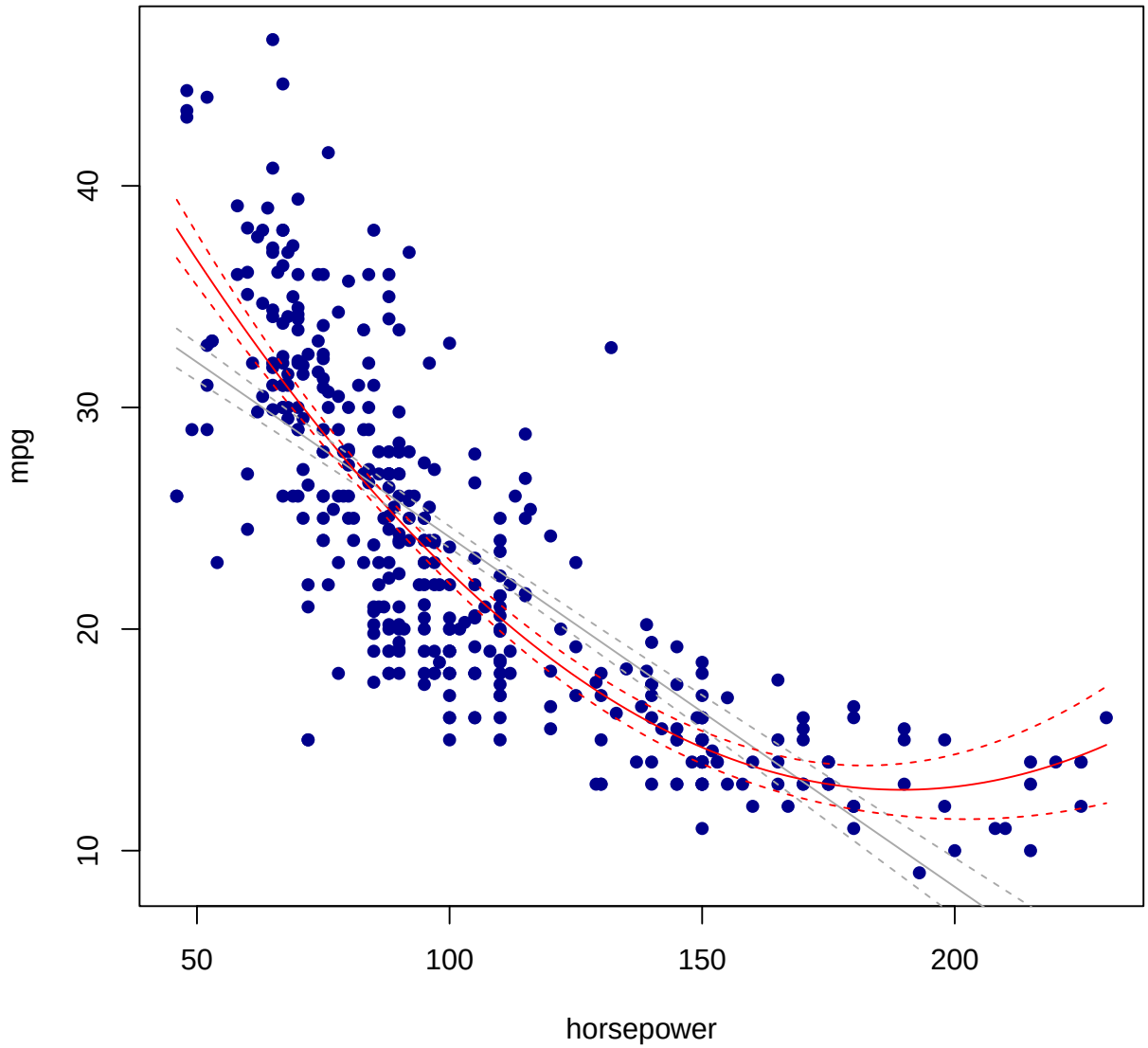
Example 5 – Auto



Example 5 – Auto

- We are interested in the relationship between miles per gallon `mpg` and the other covariates
- There are clear inverse relationships between mileage and displacement, horsepower, and weight
- However, neither of these seem to be linear, especially for American vehicles (could include interaction term!)
- A highly flexible function is unnecessary – a quadratic polynomial will suffice
- For example, first consider `horsepower`

Example 5 – Auto



Example 5 – Auto

We can clearly see that the quadratic fit (red line) captures the shape of the data better than the linear model (gray line), especially at the boundaries

However, note that we include the lower degree terms in the polynomial regression model as well – in this case the linear term alone – to capture these components in the data too

Example 5 – Auto

	Dependent variable:	
	mpg	
	(1)	(2)
horsepower	−0.158	
	p = 0.000	
poly(horsepower, 2, raw = T)1		−0.466
		p = 0.000
poly(horsepower, 2, raw = T)2		0.001
		p = 0.000
Constant	39.936	56.900
	p = 0.000	p = 0.000

Example 5 – Auto

Observations	392	392
R^2	0.606	0.688
Adjusted R^2	0.605	0.686
Residual Std. Error	4.906 (df = 390)	4.374 (df = 389)
F Statistic	599.718 (df = 1; 390)	428.018 (df = 2; 389)

Although `horsepower` is highly significant in the linear model, the quadratic fit captures a higher proportion of the variation in the data

We could add the rest of the variables in the model, although note the extreme collinearity!

Let us consider this model in the classification context

Polynomial regression – qualitative response

In the previous chapter we saw that the logistic regression model yielded linear decision boundaries since the logit is linear in $\boldsymbol{X}$

To create non-linear decision boundaries, we simply add the polynomial terms to the logit:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \cdots + \beta_d X^d, \qquad (2)$$

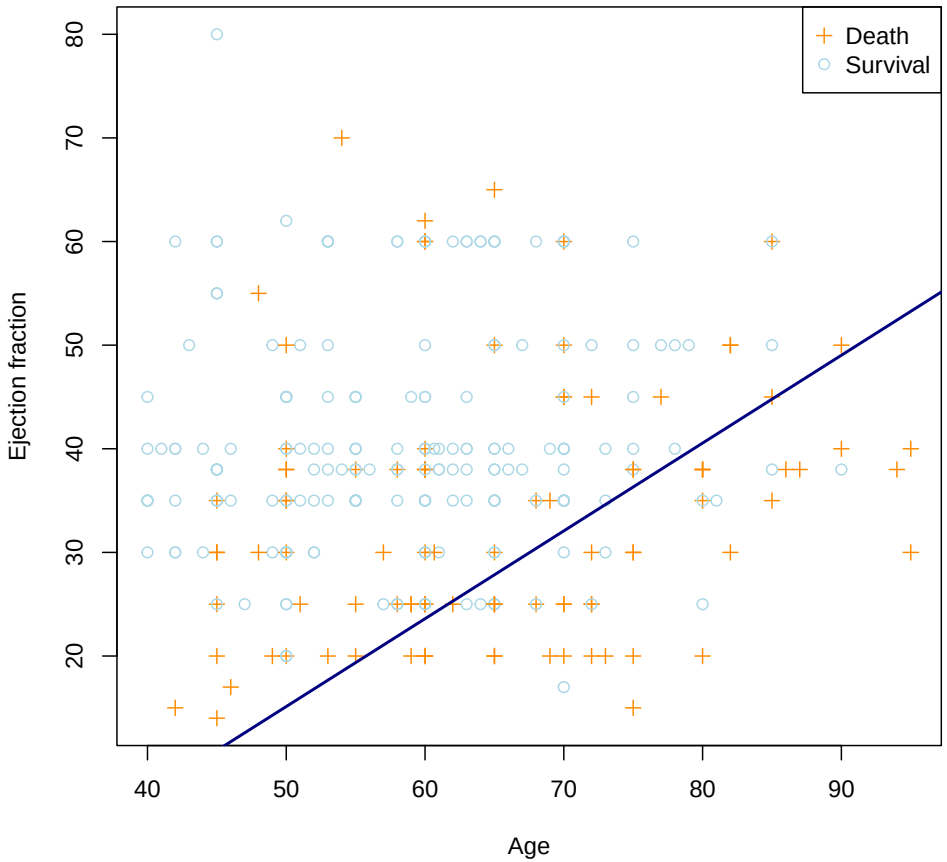
for only a single predictor X. More variables can be added to the model in a similar fashion.

To illustrate we return to the heart failure dataset

Example 4 – Heart failure (continued)

First consider only two of the numeric predictors, namely `age` and `ejection_fraction`

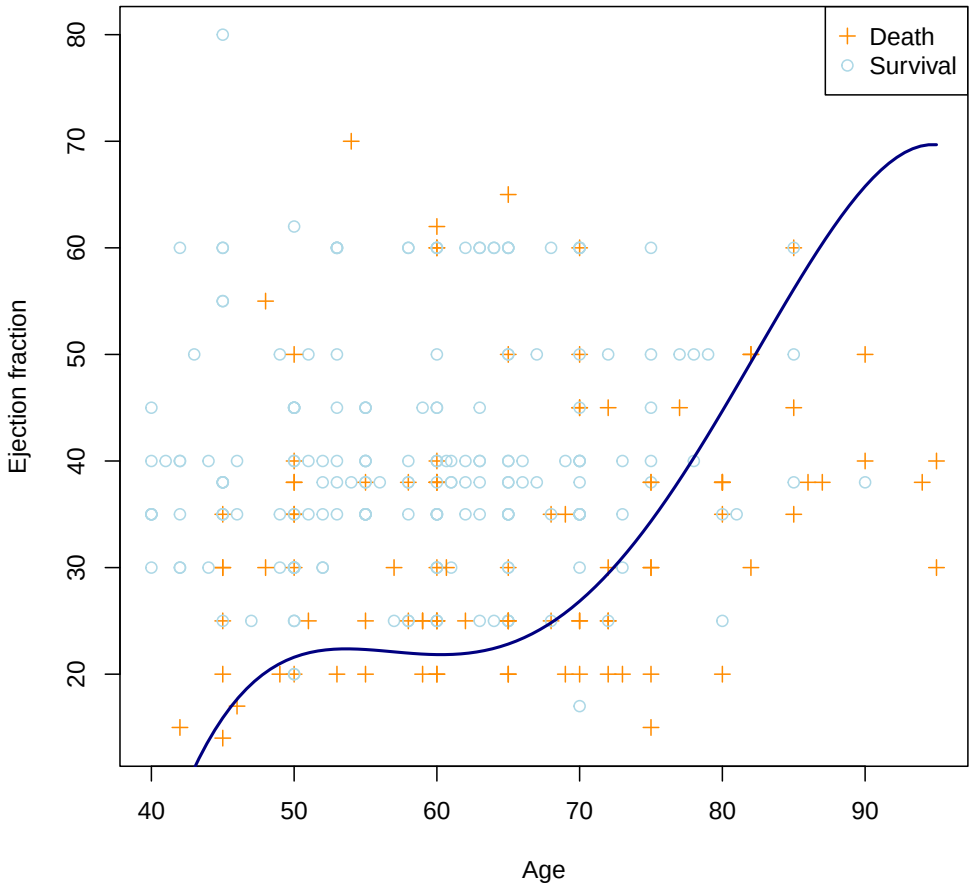
The linear logistic regression model yields the following decision boundary



Example 4 – Heart failure (continued)

Let us now propose a more flexible model:

$$\log\left(\frac{p(\boldsymbol{X})}{1-p(\boldsymbol{X})}\right) = \beta_0 + \beta_1 \text{Age} + \beta_2 \text{Age}^2 + \beta_3 \text{Age}^3 + \beta_4 \text{Age}^4 + \beta_5 \text{Eject\_Frac}$$



Example 4 – Heart failure (continued)

- Of course increased complexity does not necessarily imply improved fit or prediction!
- Need to use out-of-sample data to determine whether this model better captures the underlying relationship in the data
- Also consider what constitutes an ideal fit for the problem (weigh the asymmetric cost of misclassification) (homework!)

Basis functions and generalised additive models

- Although polynomial regression is a valuable tool for capturing non-linear relationships, it has its limitations, especially when dealing with complex data patterns.
- To address these limitations and provide more flexible modelling options, one can apply a more general **basis function** approach, where any family of functions or transformations are applied to the features.
- These functions can also be fitted piecewise (locally), defining **splines**, which are generally smoothed to be piecewise continuous and linear at the boundaries.

Basis functions and generalised additive models

- Finally, Generalised Additive Models (GAMs) are a powerful extension of linear regression that allow for the modeling of complex interactions and nonlinear relationships without relying on a single global polynomial.
- They are particularly useful when dealing with high-dimensional data and when you want to capture intricate relationships between predictors and the target.
- As with polynomial regression, these methods can be applied in both regression and classification contexts.
- Those continuing with master's will encounter these methods in advanced regression

K-Nearest Neighbours (KNN)

KNN

- K-Nearest Neighbours (KNN) is a simple non-parametric algorithm
- Can perform surprisingly well in various contexts
- KNN makes predictions based on the similarity between data points
- The premise is that similar data points tend to belong to the same class (classification) or have similar target values (regression)

KNN

- During the training phase, KNN stores the entire dataset in memory
- No explicit model is constructed and no parameters are learned – the training phase simply involves memorising the data
- When making a prediction for a new, unseen data point, KNN looks at the K nearest data points from the training dataset, where K is a user-defined **hyperparameter**
- **Euclidean distance** is generally employed as distance metric, although other measurements such as Manhattan distance and the Minkowski distance can also be used

KNN – regression

Given a value for K and a prediction point $\boldsymbol{x}_0$, KNN regression identifies the K training observations that are closest to $\boldsymbol{x}_0$.

Denote these observations by $\mathcal{N}_0$.

We then simply estimate $f(\boldsymbol{x}_0)$ as the average of the training responses in $\mathcal{N}_0$:

$$\hat{f}(\boldsymbol{x}_0) = \frac{1}{K} \sum_{\boldsymbol{x}_i \in \mathcal{N}_0} y_i. \tag{3}$$

The choice of K once again amounts to deciding on the flexibility of the decision boundary

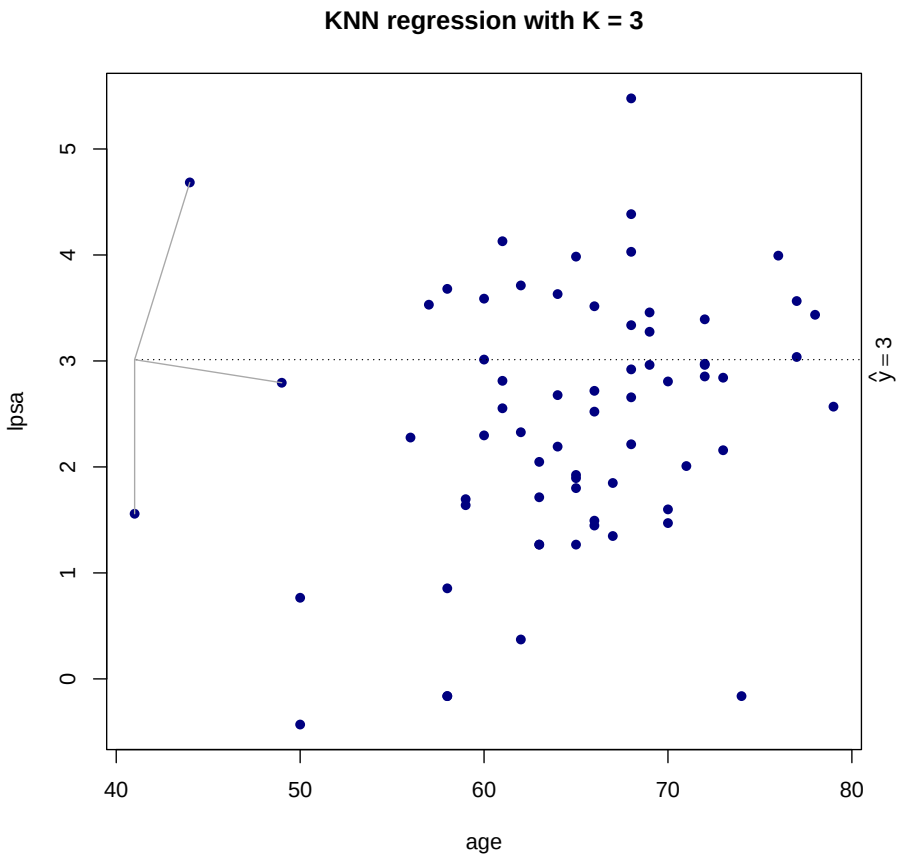
Example 2 – Prostate cancer (continued)

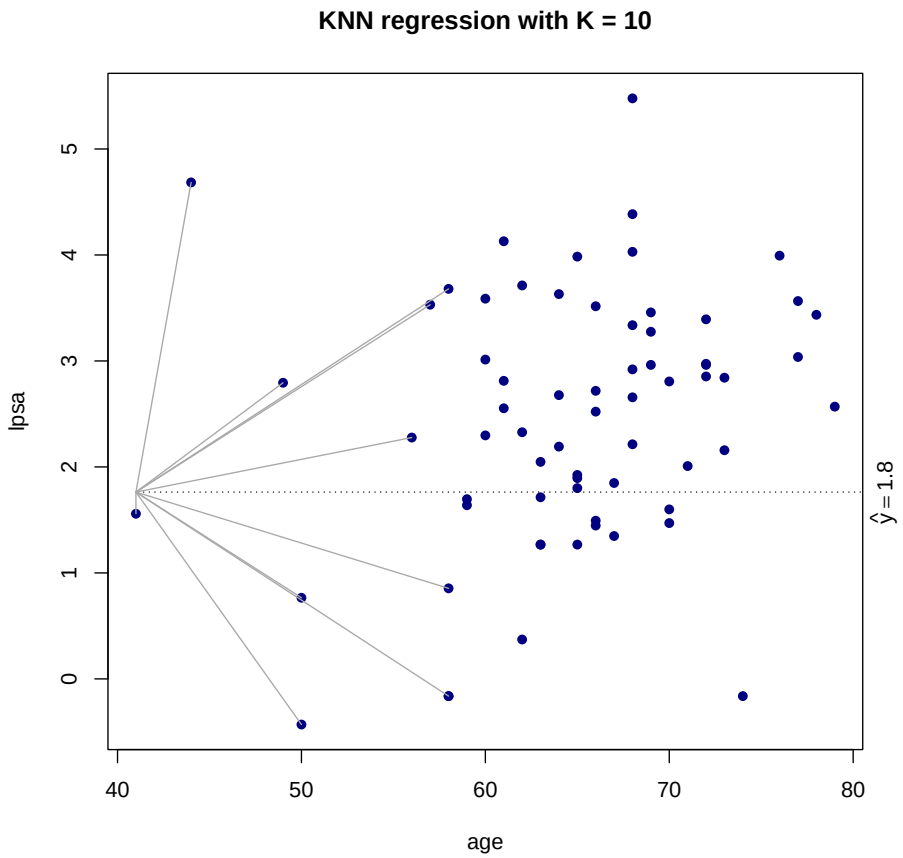
Consider again the prostate cancer dataset from Chapter 3

We will focus on the variable that was least significant in the saturated model, namely `age`

The following animation shows the fitted model across a range of ages, using both $K = 3$ and $K = 10$

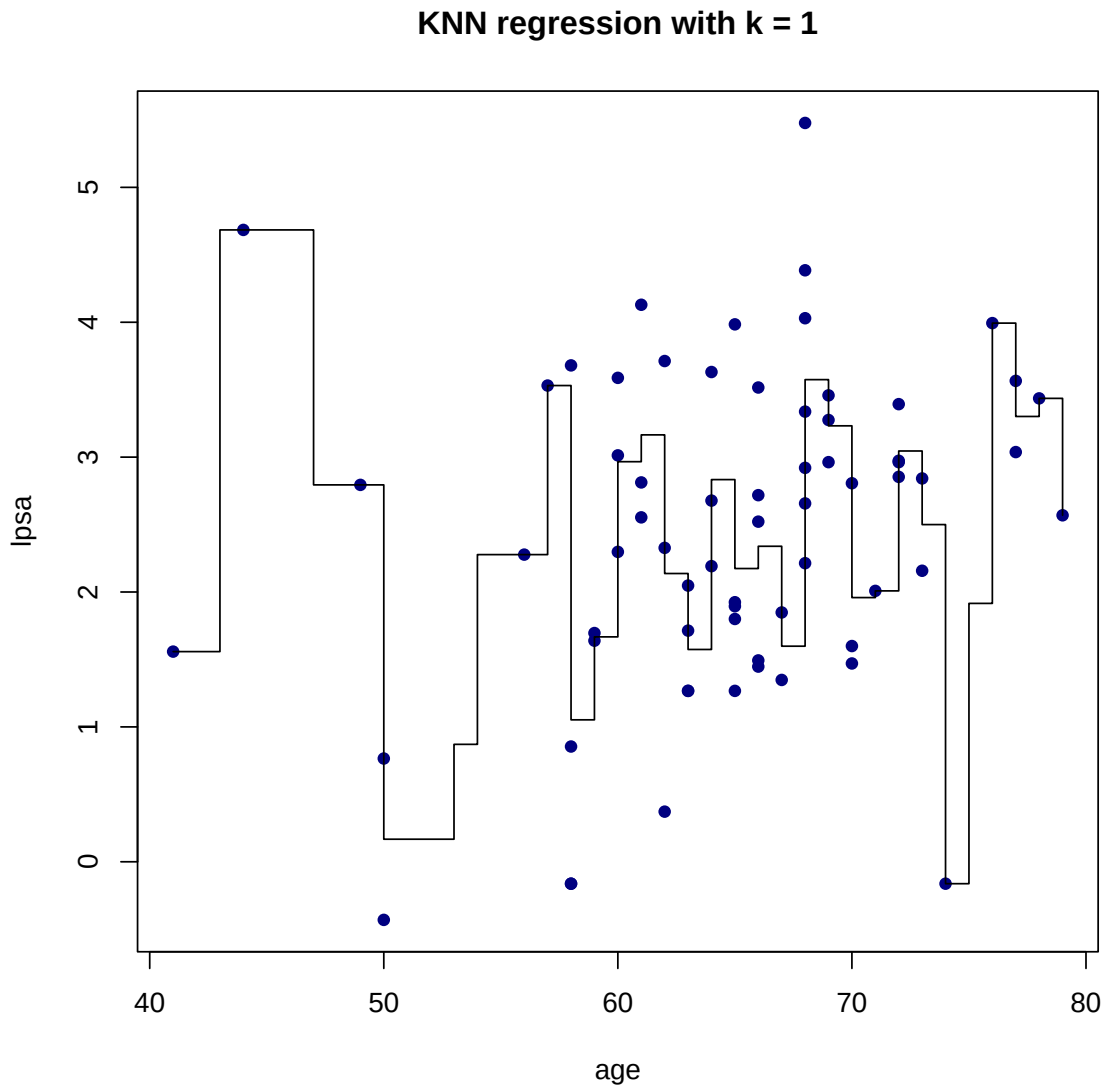
Example 2 – Prostate cancer (continued)





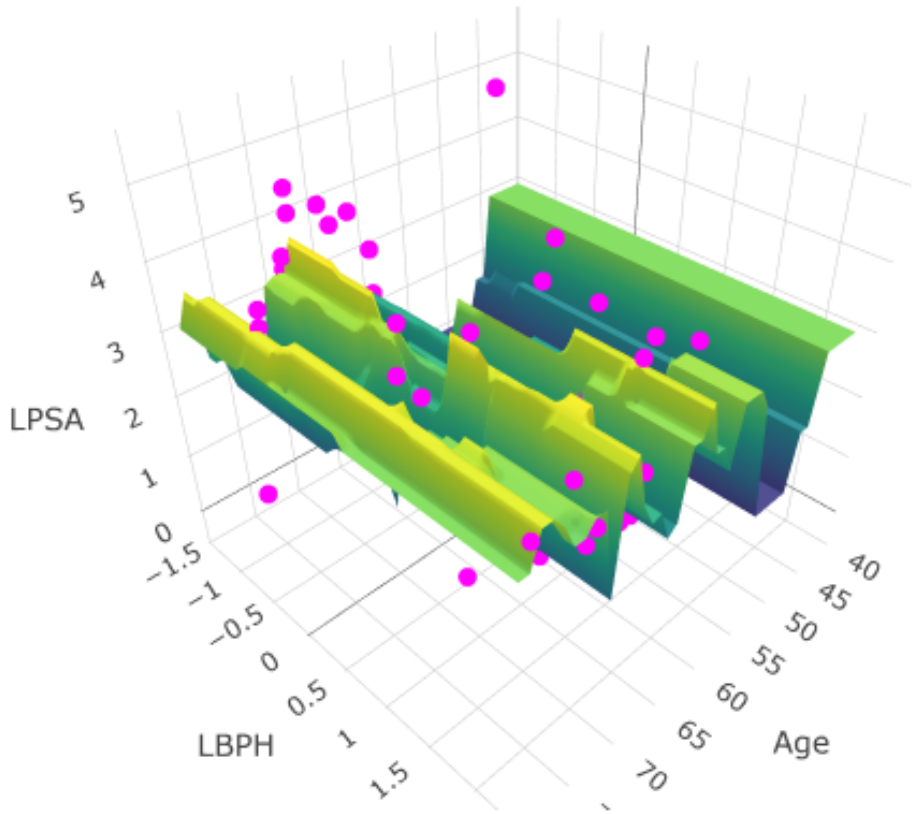
Example 2 – Prostate cancer (continued)

We observe more volatility in the fit for smaller values of K



Example 2 – Prostate cancer (continued)

When adding `lbph` to the model ($K = 3$), the stepped function becomes a stepped surface

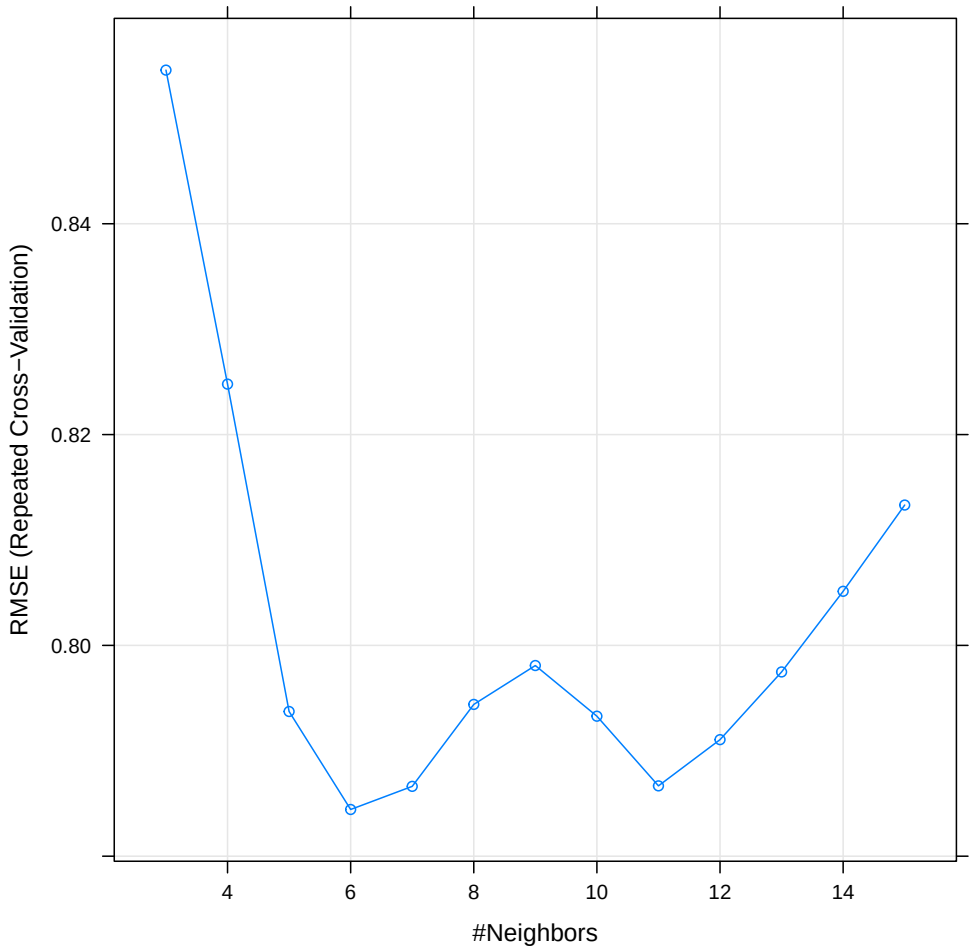


Example 2 – Prostate cancer (continued)

- Now, $K = 3$ was chosen arbitrarily here, hence the question again arises of which value of K to use
- We will again use CV to fit and validate models, varying the hyperparameter
- One of the drawbacks of KNN is that there is no sensible way of determining how much a specific variable contributes towards explaining the variance in the target variable
- Makes feature selection a difficult, often trial-and-error process
- Since we have relatively few observations in this dataset, we will only use 3 predictors – the last 3 predictors that remain in the lasso model – namely `lcavol`, `lweight`, and `svi`
- Also, since the dataset is so small, we will repeat the CV procedure 10 times and average over the results.

Example 2 – Prostate cancer (continued)

The best KNN model (lowest RMSE) has $K = 6$. Remember that lower K increases flexibility, therefore the model starts overfitting for K smaller than 6.



Example 2 – Prostate cancer (continued)

Using this model on the test set yields $MSE = 0.39$

This is actually noticeably better than our best regularised linear model, which yielded $MSE = 0.45$!

Can we improve further? Testing other combinations of features, which could possibly improve the results further, is left as a homework exercise.

Finally, we will apply this model in a classification setting

KNN – classification

The training setup in the classification setting is exactly the same as for regression

Let the target $Y \in \{1, 2, \ldots, J\}$ and again denote the K training observations closest to $\boldsymbol{x}_0$ as $\mathcal{N}_0$

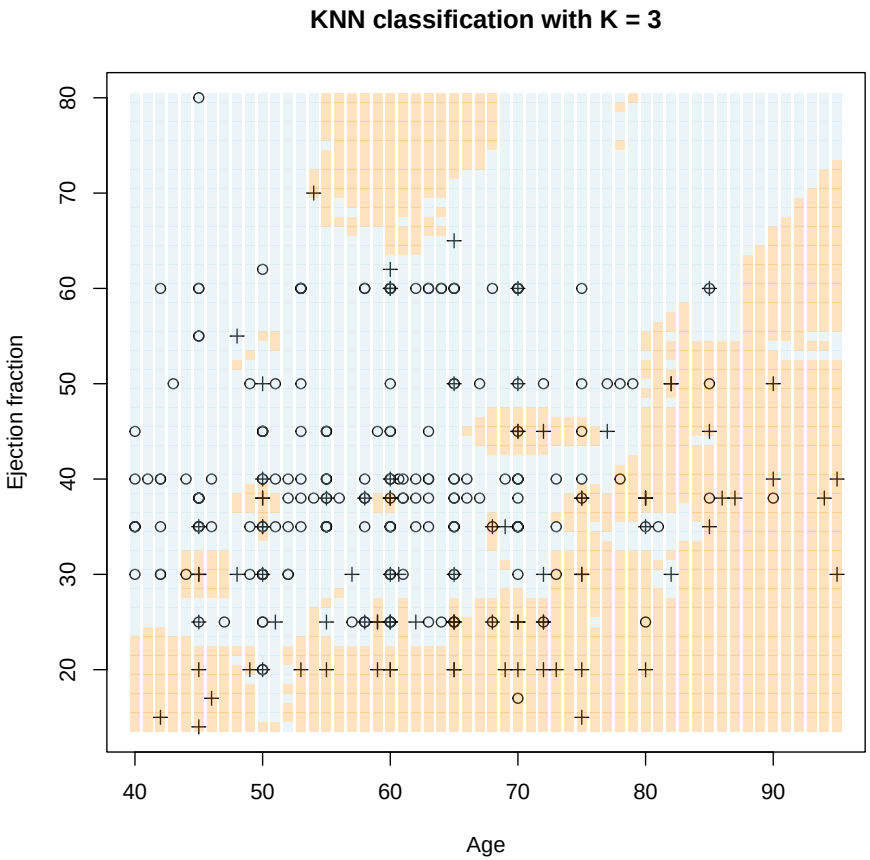
The KNN classifier then simply estimates the conditional probability for class j as the proportion of points in $\mathcal{N}_0$ whose response values equal j:

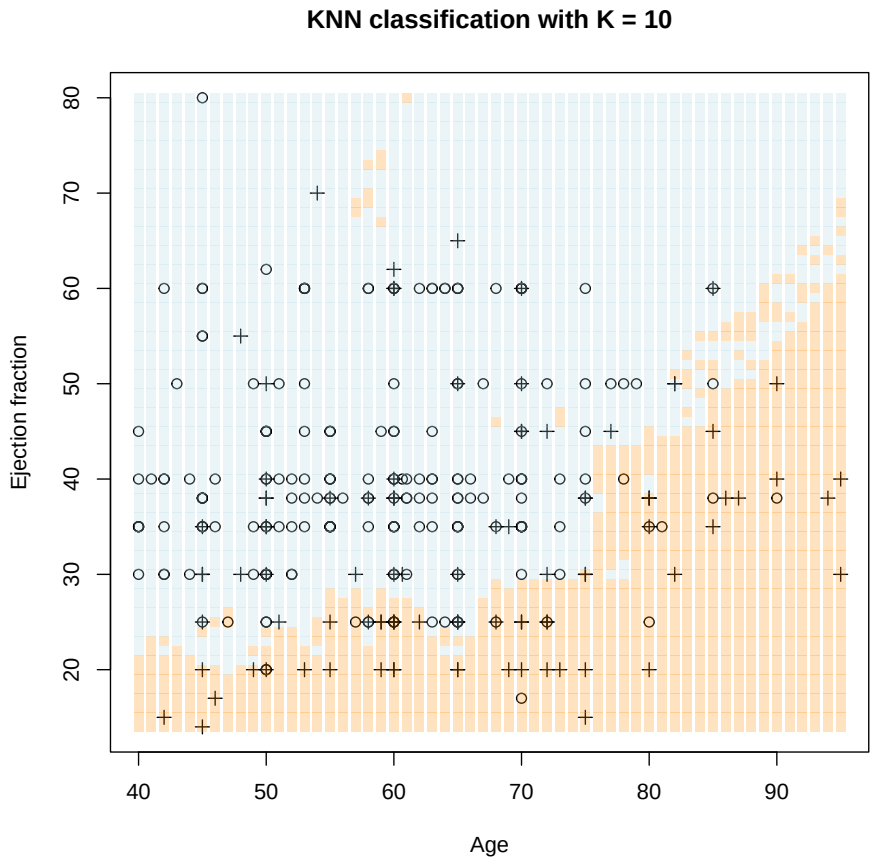
$$\Pr(Y = j | \boldsymbol{X} = \boldsymbol{x}_0) = \frac{1}{K} \sum_{\boldsymbol{x}_i \in \mathcal{N}_0} I(y_i = j). \tag{4}$$

To illustrate this, we will again return to the heart failure dataset.

Example 4 – Heart failure (continued)

To illustrate the decision boundary resulting from the KNN classifier, consider again the predictors `age` and `ejection_fraction`





Example 4 – Heart failure (continued)

- We observe highly flexible decision boundaries, which clearly fit local noise especially when K is small
- As before, we can use CV to determine the ideal complexity according to an appropriate model evaluation metric (homework!)
- To summarise, there are several advantages and disadvantages to using KNN

KNN conclusion

Advantages:

1. It is a very simple algorithm to understand and implement, for both regression and multiclass classification
2. It does not make assumptions about the decision boundaries, allowing it to capture non-linear relationships between features
3. It does not make assumptions about the distribution of the data, making it suitable for a wide range of problems

KNN conclusion

Disadvantages:

1. It requires a lot of memory and is computationally expensive for large and complex datasets
2. It is not suitable for imbalanced data (classification), as it is biased towards the majority class
3. In regression contexts it is sensitive to outliers, especially for smaller K
4. There are no neat ways of measuring variable importance or performing feature selection
5. It performs particularly poorly on very noisy data
6. It requires a lot of data for high-dimensional problems, suffering severely from the curse of dimensionality