CHAPTER 2: ABSTRACTION

REGRESSION and TREE MODELS

I. Overview

Chapter one is about "Abstraction". It concerns how we model and formulate a problem using *specific mathematical models*. Abstraction is powerful. With identification of a few main entities (usually called as variables or features) from the problem, and characterization of their relationships, we can free ourselves from the application context and focus on the study of these interconnected entities as a pure mathematical system. Consequences can be analytically (rather than speculatively) established within this abstracted framework, while phenomenon in the context could be identified as special instances of this abstracted model.

Generally, there are two main types of cultures for statistical modeling. Prof. Leo Brienman made these two cultures explicit as he articulated in his seminar paper[1]. One is the "data modeling" culture, while another one is the "algorithmic modeling" culture. In this book, we will focus on two

[1] *Leo Breiman, Statistical Modeling: The Two Cultures. Statistical Science, 2001.*

models that are representative of each culture: the linear regression models (data modeling) and decision tree models (algorithmic modeling). Linear regression is a great example about statistics-driven considerations in modeling, while decision tree is a great example about computational- and nonparametric-driven considerations in modeling.

Many real-world problems usually present themselves in the form as a mystery, as highlighted as a blackbox in Figure 2.1. In these problems, there is usually an output variable (denoted as y) we care about and want to predict; meanwhile, to help us better understand the uncertainty of the output variable, we have other variables which we call as predictors (denoted as $x_1, x_2, \ldots, x_p$). We know that there are relationships between the predictors and the output, but these relationships are unknown, due to our lack of understanding of the system. It is not always plausible or economically feasible to develop a Newtonian sytle characterization of the system using differenetial equations.

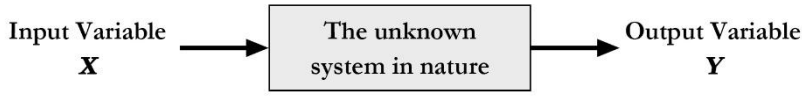

Figure 2.1: The blackbox nature of many data science problems

A common criterion for evaluating the success of any model, no matter what type of culture it belongs to, is the prediction performance on the output variable given the input variables. It is fair to say that, almost all the models in both cultures could be summarized using a generic form:

$$y = f(\boldsymbol{x}) + \epsilon,$$

where $f(\boldsymbol{x})$ reflects the deterministic part of y that can be determined by knowing $\boldsymbol{x}$, and ϵ reflects the uncertain part of y that could not be determined by $\boldsymbol{x}$ alone. In some texts, $f(\boldsymbol{x})$ is also called the model of the mean structure, i.e., since given any value of $\boldsymbol{x}$ we can predict y in the sense of an average; ϵ is usually called as the error term, noise term, or residual

term. Thus, $f(\boldsymbol{x})$ is a function of $\boldsymbol{x}$ while ϵ is usually a distribution such as Gaussian distribution with mean as zero.

With this understanding, we could summarize the different principles of both cultures in designing their belonging models:

Table 2.1: Comparison between two cultures of models

	$f(x)$	ϵ	"Cosmology"
Data Modeling	Explicit form (e.g., linear regression)	Statistical distribution (e.g., Gaussian)	Imply *Cause* and *effect*; articulate uncertainty
Algorithmic Modeling	Implicit form (e.g., tree model)	Rarely modeled as structured uncertainty; only acknowledged as meaningless noise	Look for accurate surrogate for prediction; to *fit* the data rather than to *explain* the data

II. Regression Models

II.1 Rationale and Formulation

Let's consider a simple regression model, where there is only one predictor x to predict the outcome y. Linear regression model assumes a linear form of $f(x)$, e.g.,

$$f(x) = \beta_0 + \beta_1 x,$$

and a distribution form for ϵ, e.g.,

$$\epsilon \sim N(0, \sigma_\varepsilon^2).$$

With this model, for any given value of x, we could predict the value of y as $\beta_0 + \beta_1 x$. Apparently, a few assumptions have been made:

- There is linear relationship between x and y. And this linear relationship remains the same for all the values of x. This is often referred as a global relationship between x and y. Sometimes this assumption of global relationship is too strong, e.g., as shown in

the Figure 2.2 below, in many drug research works, it is found that the dose (x) is related to the effect of the drug (y) in a varying manner that depends on the value of x. But, still, from Figure 2.2 we can also see that the linear line captures an essential component in the relationship between x and y, providing a good statistical approximation.

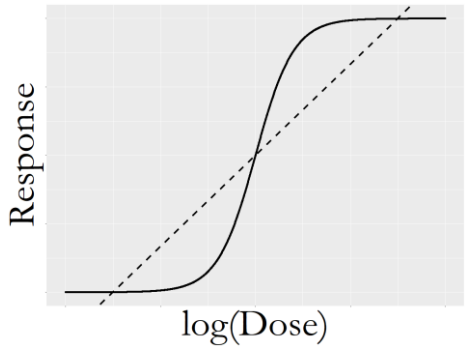

Figure 2.2: Complex relationship between dose (x) and drug response (y), while the linear line also provides a good statistical approximation

- The model suggests a fundamental unpredictability of y. That is to say, if y is generated by a combination of the signal (the $f(x)$) and the noise (ϵ), we could never predict the noise part. This has at least two implications. First, we can quantify the predictability of a dataset, by taking the ratio of $\frac{\sigma_y^2-\sigma_\varepsilon^2}{\sigma_y^2}$. Here, σ_y^2 is the overall variance of the output regardless of any predictor information. This ratio is named as ***R-squared***, that ranges from 0 (zero predictability) to 1 (perfect predictability). Second, the *significance* of x in predicting y, and the *accuracy* of x in predicting y, are two different concepts. A predictor x could be inadequate in predicting y, e.g., the R-squared could be as low as 0.1, but it still could be

statistically significant. This happens a lot in social science research and education research projects.

- The noise is usually modeled as gaussian distribution, but this assumption could be relaxed. Violation of the gaussian assumption for ε could be a concern in many applications, but not as severe as other violations such as outliers in the dataset. Of course, this assertion is empirical, only mentioned here to guide practices, and should not be taken as a strict rule.

II. 2 Theory/Method

Parameter Estimation: The regression parameters could be estimated by the least-square estimation method. A training dataset is collected to estimate the unknown parameters in the model. The basic idea is, the best parameters should fit the training data as much as possible. This is illustrated in Figure 2.3, where two principles to fit a linear regression model are shown. The vertical offsets shown in the right of Figure 2.3 is the most popular approach though. Comparing with the perpendicular offsets shown in the left of Figure 2.3, the vertical offset leads to tractability in analytic forms, which is thus more preferred.

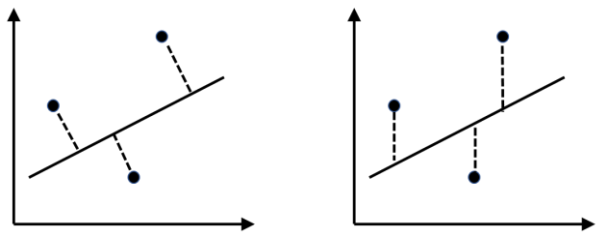

Figure 2.3: Two principles to fit a linear regression model: (left) perpendicular offsets; (right) vertical offsets.

Actually, the principle of minimizing vertical offsets leads to the least-squares estimation of linear regression models. We can exercise the least squares estimation using the simple regression model. The objective to determine the optimal line (or equivalently we can say to determine the optimal regression parameters), based on the principle suggested in the right one in Figure 2.3, is the sum of the squared of the vertical derivations of the observed data points from the line. Suppose that we have collected N data points, denoted as, (x_n, y_n) for $n = 1, 2, \dots, N$.

Then, the sum of the squared of the vertical derivations of the observed data points from the line is:

$$l(\beta_0, \beta_1) = \sum_{n=1}^{N} [y_n - (\beta_0 + \beta_1 x_n)]^2.$$

To estimate β_0 and β_1 is to minimize this least-square loss function $l(\beta_0, \beta_1)$. Thus, we could take derivatives of $l(\beta_0, \beta_1)$ regarding the two parameters and set them to be zero, to derive the estimation equations:

$$\frac{\partial l(\beta_0, \beta_1)}{\partial \beta_0} = -2 \sum_{n=1}^{N} [y_n - (\beta_0 + \beta_1 x_n)] = 0,$$

$$\frac{\partial l(\beta_0, \beta_1)}{\partial \beta_1} = -2 \sum_{n=1}^{N} x_n [y_n - (\beta_0 + \beta_1 x_n)] = 0.$$

Putting these into a succinct way, we can derive

$$\begin{bmatrix} N & \sum_{n=1}^{N} x_n \\ \sum_{n=1}^{N} x_n & \sum_{n=1}^{N} x_n^2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \sum_{n=1}^{N} y_n \\ \sum_{n=1}^{N} x_n y_n \end{bmatrix}.$$

Thus, we can solve these two equations and derive the estimator of β_0 and β_1 as

$$\beta_0 = \frac{(\sum_{n=1}^{N} y_n)(\sum_{n=1}^{N} x_n^2) - (\sum_{n=1}^{N} x_n)(\sum_{n=1}^{N} x_n y_n)}{n \sum_{n=1}^{N} x_n^2 - (\sum_{n=1}^{N} x_n)^2},$$

$$\beta_1 = \frac{\sum_{n=1}^{N} x_n y_n - N \bar{x} \bar{y}}{\sum_{n=1}^{N} x_n^2 - N \bar{x}^2}.$$

While the above mathematical expression seems to be complex, there is another angle to take a look at it. Notice that the sample correlation between x and y is:

$$cov(x, y) = \frac{\sum_{n=1}^{N} (x_n - \bar{x})(y_n - \bar{y})}{N-1} = \frac{\sum_{n=1}^{N} x_n y_n - N \bar{x} \bar{y}}{N-1},$$

Also, the sample variance is defined as

$$var(x) = \frac{\sum_{n=1}^{N} x_n^2 - N\bar{x}^2}{N-1}.$$

We can rewrite the estimators of β_0 and β_1 as

$$\beta_0 = y - \beta_1 x,$$

$$\beta_1 = \frac{cov(x,y)}{var(x)}.$$

A simple example: Let's practice the estimation method using a simple example. The dataset is shown in Table 2.2:

Table 2.2: An exemplary dataset

X	1	3	3	5	5	6	8	9
Y	2	3	5	4	6	5	7	8

The R-code to verify your calculation:

```
## Simple example of regression with one predictor
data = data.frame(rbind(c(1,2),c(3,3),c(3,5),c(5,4),c(5,6),c(6,5),
c(8,7),c(9,8)))
colnames(data) = c("Y","X")
str(data)

lm.YX <- lm(Y ~ X, data = data)
summary(lm.YX)
```

Extension to multivariate regression model: While this is the case for a simple regression model, we can extend this experience to a more general case:

$$y = \beta_0 + \sum_{i=1}^{p} \beta_i x_i + \varepsilon.$$

To fit this multivariate linear regression model, we collect n data points, denoted as

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{21} & \cdots & x_{p1} \\ 1 & x_{12} & x_{22} & \cdots & x_{p2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{1N} & x_{2N} & \cdots & x_{pN} \end{bmatrix},$$

where $\boldsymbol{y} \in R^{N\times 1}$ denotes for the n measurements of the response variable, and $\mathbf{X} \in R^{N\times(p+1)}$ denotes for the design matrix that includes the N measurements of the p input variables.

Then, the regression model can be rewritten in its matrix form as:

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}.$$

Here, $\boldsymbol{\beta} \in R^{(p+1)\times 1}$ denotes for the regression parameters and $\boldsymbol{\varepsilon} \in R^{N\times 1}$ denotes for the N residuals which are assumed to follow a normal distribution with mean as zero and variance as σ_{ε}^2.

A detailed presentation of them is shown in below:

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}, \text{ and } \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_N \end{bmatrix}.$$

Then, to estimate $\boldsymbol{\beta}$, we can derive the optimization formulation in matrix form as:

$$\min_{\boldsymbol{\beta}} (\boldsymbol{Y} - \mathbf{X}\boldsymbol{\beta})^T (\boldsymbol{Y} - \mathbf{X}\boldsymbol{\beta}),$$

To solve this optimization problem, we can take the gradient of the objective function and set it to be zero:

$$\frac{\partial (\boldsymbol{Y}-\mathbf{X}\boldsymbol{\beta})^T (\boldsymbol{Y}-\mathbf{X}\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0,$$

which gives rise to the equation:

$$\mathbf{X}^T (\boldsymbol{Y} - \mathbf{X}\boldsymbol{\beta}) = 0.$$

This leads to the least square estimator of $\boldsymbol{\beta}$ as

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{Y}.$$

A resemblance can be easily detected between $\widehat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{Y}$ with $\beta_1 = \frac{cov(x,y)}{var(x)}$ by noticing that $\mathbf{X}^T\boldsymbol{Y}$ (corresponds to $cov(x,y)$) reflects the correlation between predictors and output, and $\mathbf{X}^T\mathbf{X}$ (corresponds to $var(x)$) reflects the variability of the predictors.

Hypothesis testing of regression parameters: It is important to recognize that, since $\boldsymbol{y}$ is a random vector and induce uncertainty, $\widehat{\boldsymbol{\beta}}$ is a random vector as well. The mean of $\widehat{\boldsymbol{\beta}}$ is $\boldsymbol{\beta}$, as

$$E(\widehat{\boldsymbol{\beta}}) = E[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{y}] = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T E[\boldsymbol{y}] = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} = \boldsymbol{\beta}.$$

While the covariance matrix of $\widehat{\boldsymbol{\beta}}$ can be readily derived as

$$cov(\widehat{\boldsymbol{\beta}}) = \sigma_\varepsilon^2(\mathbf{X}^T\mathbf{X})^{-1}.$$

This result lays the foundation for developing hypothesis testing of the regression parameters.

For example, as a typical hypothesis testing question, let's say, the null hypothesis is

$$H_0: \beta_i = 0.$$

By theory, it is known that $\hat{\beta}_i \sim N\left(\beta_i, \frac{\sigma_\varepsilon^2}{\boldsymbol{x}_i^T\boldsymbol{x}_i}\right)$. Thus, if our null hypothesis is true, then, $\hat{\beta}_i \sim N\left(0, \frac{\sigma_\varepsilon^2}{\boldsymbol{x}_i^T\boldsymbol{x}_i}\right)$. This gives us the theoretical ground to make conjecture of what our estimate $\hat{\beta}_i$ is "supposed to be", i.e., as shown below, $\hat{\beta}_i$ is supposed to come from a normal distribution with mean as zero and variance as $\frac{\sigma_\varepsilon^2}{\boldsymbol{x}_i^T\boldsymbol{x}_i}$ (in a specific application, $\frac{\sigma_\varepsilon^2}{\boldsymbol{x}_i^T\boldsymbol{x}_i}$ can be calculated and take a specific value):

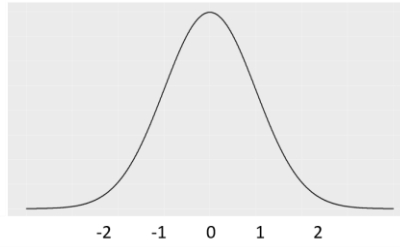

Figure 2.4: The distribution of $\hat{\beta}_i$

Based on this theory, we can see there is clearly a dominance of likelihood of what kind of $\hat{\beta}_i$ we can observe. We could define a range of $\hat{\beta}_i$ that we believe as plausible (i.e., if the null hypothesis is true, then it is normal to see this value of $\hat{\beta}_i$). Note that I use plausible in contrast with possible, since our theory tells us any value is always possible, but the possibility is not equally distributed among all the values as shown in the Figure 2.4. Also, our common sense tells us that some extreme values are always suspicious, pointing to rare chance. We may define a level of probability that represents our threshold of rare chance. We coin this threshold level as α.

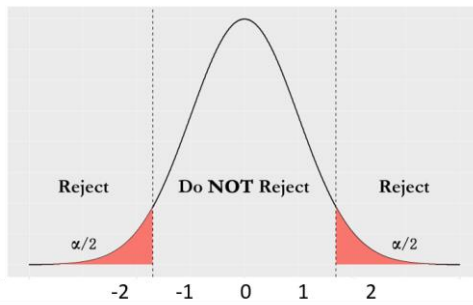

Figure 2.5: The framework of hypothesis testing

Now we have almost established the framework of hypothesis testing for regression parameters. With the threshold level α, we have *made a decision* that we will conclude that any value of $\hat{\beta}_i$ that falls into either side of the two extreme regions is unlikely – if the null hypothesis is true. Thus, if we see value in either side of the two extreme regions, we will reject the null hypothesis, since this indicates a strong conflict between theory (things suppose to be) and our empirical evidence (as what we observed on $\hat{\beta}_i$). This framework is shown in Figure 2.5.

Of course, we are conscious that we make a decision with a risk. We may be wrong, since even if the null hypothesis is true, there is still a small probability, α, that we may observe the $\hat{\beta}_i$ falls into either side of the two extreme regions. But we have accepted this risk. This risk is called the **Type 1 Error**.

II.3 R Lab

In this section, we illustrate step-by-step R codes to show how the linear regression model can be used in real-world data analysis. A distinct feature of this illustration lies on the "real-worldliness" of the data that embodies both statistical regularities (such that this analysis is enabled and called for) and realistic irregularities (such that we may recall the famous saying of Prof. George Box – "all models are wrong, some are useful"). Making informed decisions by drawing from rigorous theories, while at the same time, maintaining a critical attitude of theory, should both present simultaneously in practices of data analytics.

Here, our data is from a study of Alzheimer's disease that collected demographics information and some genetic variables from hundreds of subjects. The goal of this dataset is to use these predictors to predict the score called Mini-Mental State Examination (`MMSCORE`) which is a clinical score (from 0-30) for determining Alzheimer's disease, i.e., a `MMSCORE` of 20

to 24 suggests mild dementia, 13 to 20 suggests moderate dementia, and less than 12 indicates severe dementia.

First, let's load the data into the R workshop:

```
#### Example: Alzheimer's Disease
# filename
setwd("…/analytics/data")
AD <- read.csv('AD_bl.csv', header = TRUE)
AD$ID = c(1:dim(AD)[1])
```

It is a nice habit to make detailed documentations of the variables with R using comments:

```
# Description of variables
# ID  ID of the subjects
# Age Age
# PTGENDER Gender
# PTEDUCAT Years of education
# FDG Average FDG-PET
# AV45 Average AV45 SUVR
# HippoNV The normalized hippocampus volume
# e2_1 Apolipoprotein E4 polymorphism
# e4_1 Apolipoprotein E4 polymorphism
# rs3818361 CR1 gene rs3818361 polymorphism
# rs744373 BIN1 gene rs744373 polymorphism
# rs11136000 Clusterin CLU gene rs11136000 polymorphism
# rs610932 MS4A6A gene rs610932 polymorphism
# rs3851179 PICALM gene rs3851179 polymorphism
# rs3764650 ABCA7 gene rs3764650 polymorphism
# rs3865444 CD33 gene rs3865444 polymorphism
# MMSCORE Mini-mental state examination (outcome variable)
# TOTAL13 Alzheimer's Disease Assessment Scale (outcome variable)
```

After loading the data into the R workshop, we could use the str() function to give a sketchy overview of the data:

```
str(AD)

## 'data.frame':    517 obs. of  5 variables:
##  $ MMSCORE : int  26 30 30 28 29 30 30 27 28 30 ...
##  $ AGE     : num  71.7 77.7 72.8 69.6 70.9 65.1 79.6 73.6 60.7
 70.6 ...
##  $ PTGENDER: int  2 1 2 1 1 2 2 2 1 2 ...
##  $ PTEDUCAT: int  14 18 18 13 13 20 20 18 19 18 ...
##  $ ID      : int  1 2 3 4 5 6 7 8 9 10 ...
```

First, let's build a regression model that only uses demographics variables. Demographics variables are usually the most accessible information of patients which we can use to build prediction models.

We can create a subset of the dataset as:

```
# subset of variables we want in our first model that only uses d
emographics predictors
AD_demo <- subset(AD, select=c("MMSCORE", "AGE","PTGENDER","PTEDU
CAT","ID"))
```

Before building the model, by the spirit of **exploratory data analysis** (EDA)[1], we may draw the scatterplots to see how potentially the predictors can predict the outcome:

```
# ggplot: Plot the scatterplot of the data
# install.packages("ggplot2")
library(ggplot2)

p <- ggplot(AD_demo, aes(x = PTEDUCAT, y = MMSCORE))
p <- p + geom_point(size=2)
# p <- p + geom_smooth(method = "auto")
p <- p + labs(title="MMSE versus PTEDUCAT")
print(p)
```

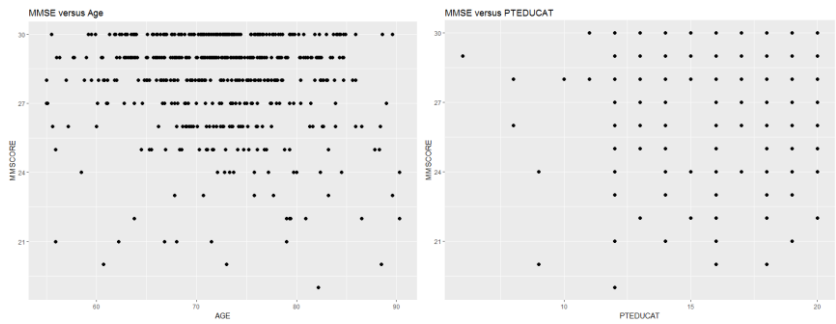

Figure 2.6: Scatterplots of (left) `MMSCORE` versus `AGE` and (right) `MMSE` versus `PTEDUCAT`

[1] John W. Tukey is a statistician whose career is known to be a strong advocate of EDA. See his book: Exploratory data analysis in 1977.

The scatterplots are shown in Figure 2.6. They show that there are weak relationships between the predictors with the `MMSCORE`, while still the relationship seems to be significant.

Then, we can use the `lm()` function to fit the regression model

```
# fit a simple linear regression model with only AGE
lm.AD_demo <- lm(MMSCORE ~ AGE, data = AD_demo)
# use summary() to get t-tests of parameters (slope, intercept)
summary(lm.AD_demo)

##
## Call:
## lm(formula = MMSCORE ~ AGE, data = AD_demo)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.7020 -0.9653  0.6948  1.6182  2.5447
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 30.44147    0.94564  32.191   <2e-16 ***
## AGE         -0.03333    0.01296  -2.572   0.0104 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.11 on 515 degrees of freedom
## Multiple R-squared:  0.01268,    Adjusted R-squared:  0.01076
## F-statistic: 6.614 on 1 and 515 DF,  p-value: 0.0104
```

Some important details could be read from the results shown above. First, it can be seen that the predictor, `AGE`, is significant with `p-value` as `0.0104` by the hypothesis testing procedure we delineated in Section II.2. It also seems that, the effect of this predictor, comparing with the noise, is rather weak, as the `R-squared` is only `0.01268`, suggesting that only 1.2% of the variability in `MMSCORE` could be explained by `AGE` alone.

To increase the `R-squared`, now let's include all the demographics variables into the model:

```
# fit the multiple linear regression model with more than one pre
dictor
lm.AD_demo2 <- lm(MMSCORE ~ AGE + PTGENDER + PTEDUCAT, data = AD_
demo)
summary(lm.AD_demo2)
```

```
##
## Call:
## lm(formula = MMSCORE ~ AGE + PTGENDER + PTEDUCAT, data = AD_de
mo)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.4290 -0.9766  0.5796  1.4252  3.4539
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 27.70377    1.11131  24.929  < 2e-16 ***
## AGE         -0.02453    0.01282  -1.913   0.0563 .
## PTGENDER    -0.43356    0.18740  -2.314   0.0211 *
## PTEDUCAT     0.17120    0.03432   4.988 8.35e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.062 on 513 degrees of freedom
## Multiple R-squared:  0.0612, Adjusted R-squared:  0.05571
## F-statistic: 11.15 on 3 and 513 DF,  p-value: 4.245e-07
```

From the results shown above we can see that, the predictor `AGE` is now on the boardline of significance with a `p-value` as `0.0563`. The other predictors, `PTGENDER` and `PTEDUCAT`, are significant. It also seems that the `R-squared` now increases from `0.01268` to `0.0612`, suggesting that 6.12% of the variability in `MMSCORE` could be explained by the three variables. The reason that the predictor `AGE` is now no longer significant is an interesting phenomena, but it is not unusual that a significant predictor becomes insignificant when other variables are included or excluded. This is because of the statistical dependence of the estimation of the predictors. As we have known that

$$cov(\widehat{\boldsymbol{\beta}}) = \sigma_{\varepsilon}^{2}(\mathbf{X}^{T}\mathbf{X})^{-1}.$$

As long as $\mathbf{X}^{T}\mathbf{X}$ is not an identify matrix, the estimators of the regression parameters are dependent in a complicated and data-driven way. Due to this reason, we need to be very cautious about how to interpret the regression parameters as they are actually interrelated and also depend on the modeling process.

Having said that, regression model is still a useful approach to provide prediction power and insights about the data. Now let's build a full model with all the demographics, genetics, and imaging variables to predict `MMSCORE`.

```
# fit a full-scale model
AD_full <- AD[,c(1:16)]
lm.AD <- lm(MMSCORE ~ ., data = AD_full)

summary(lm.AD)

##
## Call:
## lm(formula = MMSCORE ~ ., data = AD_full)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3201 -1.0265  0.2765  1.1977  4.1463
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 18.09118    1.69735  10.659  < 2e-16 ***
## AGE          0.01365    0.01197   1.140 0.254794
## PTGENDER    -0.13146    0.16356  -0.804 0.421944
## PTEDUCAT     0.16020    0.02947   5.436 8.53e-08 ***
## FDG          0.86143    0.13368   6.444 2.74e-10 ***
## AV45        -1.55526    0.42909  -3.625 0.000319 ***
## HippoNV      7.27789    1.20610   6.034 3.11e-09 ***
## e2_1        -0.03103    0.27459  -0.113 0.910068
## e4_1        -0.18525    0.17651  -1.049 0.294456
## rs3818361    0.18737    0.16373   1.144 0.253007
## rs744373    -0.30165    0.15576  -1.937 0.053359 .
## rs11136000  -0.03018    0.16423  -0.184 0.854257
## rs610932    -0.34879    0.16208  -2.152 0.031872 *
## rs3851179    0.05742    0.15675   0.366 0.714276
## rs3764650    0.31522    0.19691   1.601 0.110049
## rs3865444   -0.38589    0.15474  -2.494 0.012960 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.744 on 501 degrees of freedom
## Multiple R-squared:  0.3443, Adjusted R-squared:  0.3246
## F-statistic: 17.54 on 15 and 501 DF,  p-value: < 2.2e-16
```

From the results shown above we can see that, the `PTEDUCAT` `FDG`, AV45, `HoppoNV`, `rs610932`, and `rs3865444`, are significant. It also seems that the `R-`

`squared` now increases from `0.0612`to `0.3443`, suggesting that now 33.43% of the variability in `MMSCORE` could be explained by the variables.

We also notice that there are many variables showing insignificant p-values. Thus, we may conduct a **feature selection** procedure to delete the insignificant variables from the model. Roughly speaking, there are two approaches. One is called **stepwise backward** that begins with a full model, and sequentially remove variables. Another approach is called the **stepwise foreward**, that begins with a one-predictor model and then sequentially adds variables into the model. Here, let's use the stepwise backward method for an example to show how it works.

The first variable to be removed from the model is the one that has the largest `p-value` (thus least significant).

```
# Do we need all the variables?
# remove e2_1, as it is least significant
lm.AD.reduced <- lm.AD;
lm.AD.reduced <- update(lm.AD.reduced, ~ . - e2_1);
summary(lm.AD.reduced);

##
## Call:
## lm(formula = MMSCORE ~ AGE + PTGENDER + PTEDUCAT + FDG + AV45
+
##     HippoNV + e4_1 + rs3818361 + rs744373 + rs11136000 + rs610
932 +
##     rs3851179 + rs3764650 + rs3865444, data = AD_full)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3189 -1.0216  0.2807  1.2016  4.1466
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 18.08148    1.69351  10.677  < 2e-16 ***
## AGE          0.01367    0.01196   1.143 0.253499
## PTGENDER    -0.13191    0.16335  -0.808 0.419758
## PTEDUCAT     0.16022    0.02944   5.442 8.25e-08 ***
## FDG          0.86185    0.13350   6.456 2.54e-10 ***
## AV45        -1.55316    0.42826  -3.627 0.000316 ***
## HippoNV      7.27258    1.20400   6.040 3.00e-09 ***
## e4_1        -0.18202    0.17401  -1.046 0.296053
## rs3818361    0.18809    0.16345   1.151 0.250379
## rs744373    -0.30116    0.15555  -1.936 0.053417 .
```

```
## rs11136000  -0.03037    0.16406  -0.185 0.853200
## rs610932    -0.34840    0.16188  -2.152 0.031854 *
## rs3851179    0.05936    0.15565   0.381 0.703078
## rs3764650    0.31553    0.19670   1.604 0.109322
## rs3865444   -0.38599    0.15459  -2.497 0.012848 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.742 on 502 degrees of freedom
## Multiple R-squared:  0.3443, Adjusted R-squared:  0.326
## F-statistic: 18.82 on 14 and 502 DF,  p-value: < 2.2e-16
```

It can be seen that the `R-squared` is not affected. To formally draw a conclusion, we can compare the full model with this new model by `F-test` that is implemented in `anova()`:

```
anova(lm.AD.reduced,lm.AD)

## Analysis of Variance Table
##
## Model 1: MMSCORE ~ AGE + PTGENDER + PTEDUCAT + FDG + AV45 + Hi
ppoNV +
##     e4_1 + rs3818361 + rs744373 + rs11136000 + rs610932 + rs38
51179 +
##     rs3764650 + rs3865444
## Model 2: MMSCORE ~ AGE + PTGENDER + PTEDUCAT + FDG + AV45 + Hi
ppoNV +
##     e2_1 + e4_1 + rs3818361 + rs744373 + rs11136000 + rs610932
 +
##     rs3851179 + rs3764650 + rs3865444
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    502 1523.2
## 2    501 1523.1  1  0.038826 0.0128 0.9101
```

And we can see that it is statistically indistinguishable between the two models by the `F-test`, with `p-value` as `0.9101`.

We then move forward to delete the latest least significant predictor, `rs11136000`:

```
lm.AD.reduced <- update(lm.AD.reduced, ~ . - rs11136000);
summary(lm.AD.reduced);

##
## Call:
## lm(formula = MMSCORE ~ AGE + PTGENDER + PTEDUCAT + FDG + AV45
+
```

```
##     HippoNV + e4_1 + rs3818361 + rs744373 + rs610932 + rs38511
79 +
##     rs3764650 + rs3865444, data = AD_full)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3315 -1.0138  0.2713  1.1929  4.1375
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 18.05037    1.68353  10.722  < 2e-16 ***
## AGE          0.01360    0.01194   1.139 0.255316
## PTGENDER    -0.13228    0.16318  -0.811 0.417982
## PTEDUCAT     0.16035    0.02941   5.453 7.78e-08 ***
## FDG          0.86249    0.13333   6.469 2.34e-10 ***
## AV45        -1.54367    0.42477  -3.634 0.000308 ***
## HippoNV      7.26894    1.20268   6.044 2.93e-09 ***
## e4_1        -0.18292    0.17377  -1.053 0.293003
## rs3818361    0.19161    0.16218   1.181 0.237973
## rs744373    -0.30130    0.15540  -1.939 0.053077 .
## rs610932    -0.34802    0.16171  -2.152 0.031863 *
## rs3851179    0.06092    0.15527   0.392 0.694989
## rs3764650    0.31577    0.19651   1.607 0.108700
## rs3865444   -0.38681    0.15437  -2.506 0.012536 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.74 on 503 degrees of freedom
## Multiple R-squared:  0.3442, Adjusted R-squared:  0.3273
## F-statistic: 20.31 on 13 and 503 DF,  p-value: < 2.2e-16
```

Again, we can compare the full model with this new model by `F-test`

```
anova(lm.AD.reduced,lm.AD)

## Analysis of Variance Table
##
## Model 1: MMSCORE ~ AGE + PTGENDER + PTEDUCAT + FDG + AV45 + Hi
ppoNV +
##     e4_1 + rs3818361 + rs744373 + rs610932 + rs3851179 + rs376
4650 +
##     rs3865444
## Model 2: MMSCORE ~ AGE + PTGENDER + PTEDUCAT + FDG + AV45 + Hi
ppoNV +
##     e2_1 + e4_1 + rs3818361 + rs744373 + rs11136000 + rs610932
 +
##     rs3851179 + rs3764650 + rs3865444
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
```

```
## 1    503 1523.2
## 2    501 1523.1  2   0.14282 0.0235 0.9768
```

We can repeat this process, until no more variable could be deleted.

While this approach is simple and gives us great visibility of the model selection process, it is a tedious process. Automation of this process could be achieved by the function `step()`, as shown in below:

```
# Automatic model selection
lm.AD.F <- step(lm.AD, direction="backward", test="F")
```

Then we can obtain the final selected model as:

```
## Step:  AIC=581.47
## MMSCORE ~ PTEDUCAT + FDG + AV45 + HippoNV + rs744373 + rs61093
2 +
##     rs3764650 + rs3865444
##
##             Df Sum of Sq    RSS    AIC F value    Pr(>F)
## <none>                   1537.5 581.47
## - rs3764650  1     7.513 1545.0 581.99  2.4824  0.115750
## - rs744373   1    12.119 1549.6 583.53  4.0040  0.045924 *
## - rs610932   1    14.052 1551.6 584.17  4.6429  0.031652 *
## - rs3865444  1    21.371 1558.9 586.61  7.0612  0.008125 **
## - AV45       1    50.118 1587.6 596.05 16.5591 5.467e-05 ***
## - PTEDUCAT   1    82.478 1620.0 606.49 27.2507 2.610e-07 ***
## - HippoNV    1   118.599 1656.1 617.89 39.1854 8.206e-10 ***
## - FDG        1   143.852 1681.4 625.71 47.5288 1.614e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It can be seen that the predictors kept in the final model are all significant. Also, the `R-squared` is `0.3381` using the 8 predictors. This is not bad comparing with the R-squared as `0.3443` by using all the 15 predictors. Again, we can compare the full model with this final model by `F-test`, which shows that the it is statistically indistinguishable between the two models by the `F-test`, with `p-value` as `0.6913`.

```
anova(lm.AD.F,lm.AD)

## Analysis of Variance Table
##
## Model 1: MMSCORE ~ PTEDUCAT + FDG + AV45 + HippoNV + rs744373
+ rs610932 +
```

```
##     rs3764650 + rs3865444
## Model 2: MMSCORE ~ AGE + PTGENDER + PTEDUCAT + FDG + AV45 + Hi
ppoNV +
##     e2_1 + e4_1 + rs3818361 + rs744373 + rs11136000 + rs610932
 +
##     rs3851179 + rs3764650 + rs3865444
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    508 1537.5
## 2    501 1523.1  7    14.414 0.6773 0.6913
```

II.4 Remarks

The regression model is a very useful model, while at the same time, it is a model that demands a great amount of meticulous analysis and justification. It also a model that is frequently reported to be a troublemaker for confusion or misinterpretation, while a common misinterpretation is to treat the *statistical significance* of a predictor as a *causal significance* in the application context. Indeed, it is tempting to ignore the statistical nature of the regression model and treat it as a causal model given its asymmetric form (i.e., predictors are in one side of equation while outcome is in the other side). As we have seen in our example, some variables showing significance may disappear when other variables are added into the model, providing empirical evidences that the regression model is essentially a hypothesized model, whose statistical validity relies on its goodness-of-fit on the data.

A model fits the data well and passes the significance test only means that in data there is nothing significant against the model, but this fit process doesn't mean we found in data that this model is the only causal model that excludes the possibility of other models.

Related to this, as we briefly mentioned before, the design of experiment (DOE) is a discipline which tries to provide systematic data collection procedure to render the regression model as a causal inference model. How this could be done demands a lengthy discussion and illustration. Here, we briefly review its foundation to see why it has the connection with linear regression model.

It can be seen that, the uncertainty of $\widehat{\boldsymbol{\beta}}$ mainly comes from two sources, the variability from the data that is encoded in σ_ε^2, and the structure of $\mathbf{X}$. The noise encoded in σ_ε^2 reflects essential uncertainty inherent in the system or the measurement device that generates the data. But $\mathbf{X}$ is how we collect the data at what data points. Thus, many experimental design methods seek to optimize the structure of $\mathbf{X}$. We can try different cases of the matrix $\mathbf{X}$ to see the implication of this result.

For example, let's consider the following $\mathbf{X}$:

$$\mathbf{X} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

It can be seen that, given this structure, the variance of $\widehat{\boldsymbol{\beta}}$ takes a very special structure:

$$cov(\widehat{\boldsymbol{\beta}}) = \sigma_\varepsilon^2 \boldsymbol{I}.$$

In other words, we can draw two main observations. First, the estimations of the regression parameters are independent, given that their correlations are zero. Second, the variances of the regression parameters are the same. From these two traits, this is an ideal structure for the data matrix $\mathbf{X}$ that is commonly adopted for design of experiments when we have control over what data points we could collect. On the other hand, the data matrix $\mathbf{X}$ most often takes an arbitrary structure that results in a general form for $cov(\widehat{\boldsymbol{\beta}})$. Thus, estimations of the regression parameters are often correlated with each other. Adding or deleting variables from the regression model will result in changes of the estimations of other parameters, calling for cautions from us to interpret the regression models properly.

Another interesting remark we'd like to point it out is that, in regression models, it is often the case that the interactions of the predictors could contribute extra prediction power. For example, to predict the yield of a chemical production process using the predictors, Temperature and

Catalyst concentration, it is likely that we need to include the main effects of both predictors, but also their interactions. But, generally speaking, it is challenging in practice to recognize that there are important interaction terms to be included in the model. Thus, we need both contextual knowledge and meticulous craftwork of analytics to play with the data and interrogate the data.

Here we provide an exemplary illustration of how to play with the data using EDA to discover interactions among variables. Thinking of the interaction between two variables, let's say, X_1 and X_2, it essentially suggests that the relationship between X_1 (or X_2) and the outcome Y depends on what value the other variable X_2 (or X_1) takes. Thus, this gives us an insight that, as we can use scatterplot to visualize the relationship between any variable with the outcome, we could see how this relationship changes according to another variable.

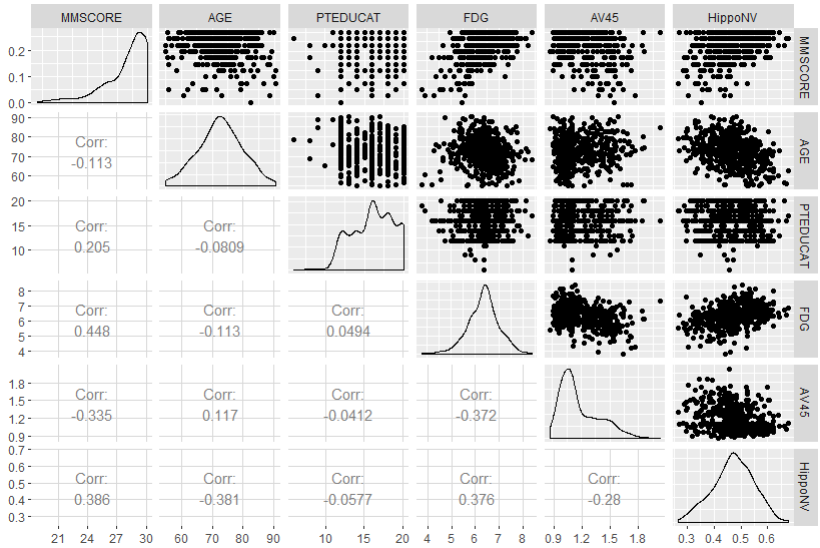

Figure 2.7: Scatterplots of the continuous predictors versus outcome variable

To implement this idea, first, let's use the AD dataset, and draw the scatterplot of the predictors as shown in Figure 2.7.

```
# Supplement the model with some visualization of the statistical
 patterns
# Scatterplot matrix to visualize the relationship between outcom
e variable with continuous predictors
library(ggplot2)
# install.packages("GGally")
library(GGally)

# draw the scatterplots and also empirical shapes of the distribu
tions of the variables
p <- ggpairs(AD[,c(16,1,3,4,5,6)], upper = list(continuous = "poi
nts")
             , lower = list(continuous = "cor")
)
print(p)
```

For the other predictors which are binary, we can use boxplot, which is shown in Figure 2.8.

```
# Boxplot to visualize the relationship between outcome variable
with categorical predictors
library(ggplot2)
qplot(factor(PTGENDER), MMSCORE, data = AD,
      geom=c("boxplot"), fill = factor(PTGENDER))
```

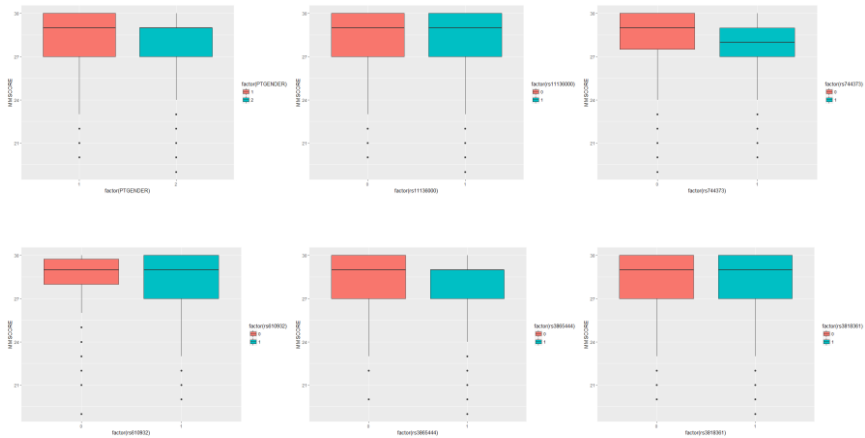

Figure 2.8: Boxplots of the binary predictors versus outcome variable

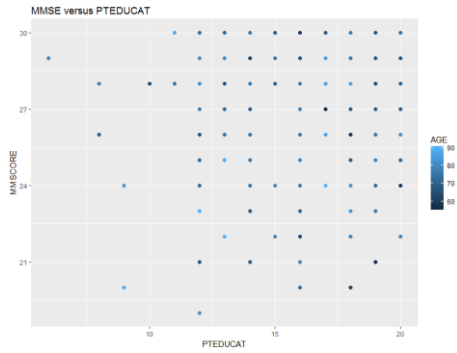

Figure 2.9: Scatterplots of PTEDUCAT versus MMSE

Now, let's pick up the scatterplot of `MMSCORE` versus `PTEDUCAT`, and see if the predictor, `AGE`, mediates the relationship between `MMSCORE` and `PTEDUCAT`. We then color the data points in the scatterplot while the color corresponds to the numerical scale of `AGE`. The following R codes generate Figures 2.9 and 2.10.

```
# How to detect interaction terms by exploratory data analysis (E
DA)
require(ggplot2)
p <- ggplot(AD_demo, aes(x = PTEDUCAT, y = MMSCORE))
p <- p + geom_point(aes(colour=AGE), size=2)
# p <- p + geom_smooth(method = "auto")
p <- p + labs(title="MMSE versus PTEDUCAT")
print(p)
```

It looks like that the relationship between `MMSCORE` and `PTEDUCAT` indeed changes according to different levels of `AGE`. Thus, we draw the same scatterplot on two levels of `AGE`, `AGE < 60` and `AGE > 80`.

```
p <- ggplot(AD_demo[which(AD_demo$AGE < 60),], aes(x = PTEDUCAT,
y = MMSCORE))
p <- p + geom_point(size=2)
p <- p + geom_smooth(method = lm)
p <- p + labs(title="MMSE versus PTEDUCAT when AGE < 60")
print(p)

p <- ggplot(AD_demo[which(AD_demo$AGE > 80),], aes(x = PTEDUCAT,
y = MMSCORE))
p <- p + geom_point(size=2)
p <- p + geom_smooth(method = lm)
```

```
p <- p + labs(title="MMSE versus PTEDUCAT when AGE > 80")
print(p)
```

Then, we can obtain Figure 2.10.

Obviously, an interesting phenomenon emerges and shows that the relationship between `MMSCORE` and `PTEDUCAT` changes dramatically according to different levels of `AGE`!

Thus, we further add this interaction term into the model that uses all the demographics variables:

```
# fit the multiple linear regression model with an interaction te
rm: AGE*PTEDUCAT
lm.AD_demo2 <- lm(MMSCORE ~ AGE + PTGENDER + PTEDUCAT + AGE*PTEDU
CAT, data = AD_demo)
summary(lm.AD_demo2)
```

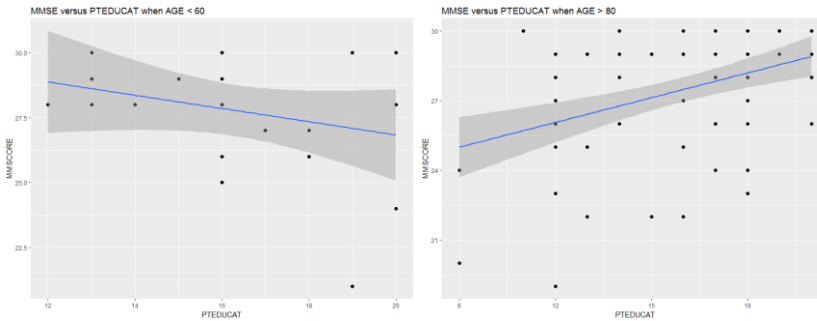

Figure 2.10: Scatterplots of `PTEDUCAT` versus `MMSCORE` when (left) `AGE < 60` or (right) `AGE > 80`

Then, we can see that this interaction term is significantly contributing extra prediction power on top of the existing predictors (`p-value` is `0.01534`)!

```
##
## Call:
## lm(formula = MMSCORE ~ AGE + PTGENDER + PTEDUCAT + AGE * PTEDU
CAT,
##     data = AD_demo)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -8.2571 -0.9204  0.5156  1.4219  4.2975
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  40.809411   5.500441   7.419 4.93e-13 ***
## AGE          -0.202043   0.074087  -2.727  0.00661 **
## PTGENDER     -0.470951   0.187143  -2.517  0.01216 *
## PTEDUCAT     -0.642352   0.336212  -1.911  0.05662 .
## AGE:PTEDUCAT  0.011083   0.004557   2.432  0.01534 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.052 on 512 degrees of freedom
## Multiple R-squared:  0.07193,    Adjusted R-squared:  0.06468
## F-statistic:  9.92 on 4 and 512 DF,  p-value: 9.748e-08
```

III. Tree Models

III.1 Rationale and Formulation

While the linear regression model is a typical data modeling method, the decision tree model represents a typical method in the category of algorithmic modeling as discussed in Table 1. The linear regression model formalizes the data generating mechanism, which emphasizes an understanding of the underlying system. In constrat, the decision tree mimics heuristics in human reasoning. An exemplary tree model is shown in Figure 2.11, which uses weather and day of week (Dow) (as two predictors) to predict whether to play basketball (as outcome variable).

As shown in Figure 2.11, a decision tree contains of one root node (highlighted as yellow), inner nodes (highlighted as yellow), and decision nodes (highlighted as green). It also has splitting rules that are specified alongside the arcs. To predict on a data point, i.e., $\boldsymbol{x_i}$, the root node is where to begin with. The data point will travel along the inner nodes according to the rules specified alongside the arcs. For example, considering the tree model in Figure 2.11. If $\boldsymbol{x_i} = \{\text{Weather} = \text{Sunny}, \text{Dow} = \text{Yes}\}$, then we can see that the data point will first go to inner node 1, and then, go to the left decision node and reach the decision that Play = Yes. If $\boldsymbol{x_i} = \{\text{Weather} != \text{Sunny}, \text{Dow} = \text{Yes}\}$, then we can see that the data point

will go to the decision node right from the root node, and reach the decision that Play = No.

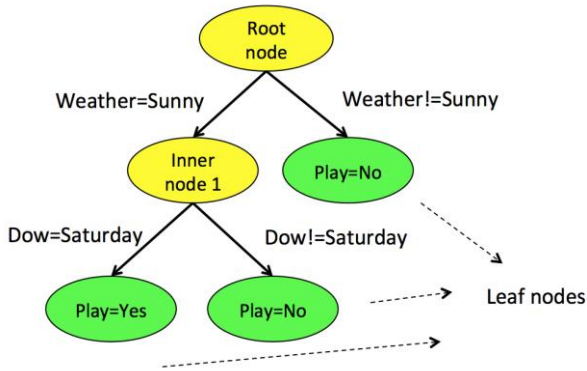

Figure 2.11: An exemplary decision tree model

Table 2.3: An exemplary dataset for building a decision tree

ID	Weather	Dow (day-of-week)	Play
1	Rainy	Saturday	No
2	Sunny	Saturday	Yes
3	Windy	Tuesday	No
4	Sunny	Saturday	Yes
5	Sunny	Monday	No
6	Windy	Saturday	No

III.2 Theory/Method

The decision tree shown in Figure 2.11 is created by domain knowledge. In what follows, we introduce how to create such a decision tree using data. As we can see from Figure 2.11, the key elements of a decision tree is the splitting rules that can lead us through the inner nodes to the final decision node to reach a decision. A splitting rule is defined by a variable and the set

of values it belongs to, e.g., Weather = Sunny. The variable used for splitting is referred as the splitting variable, and the corresponding value is refereed as the splitting value.

Pretending that we don't have the domain knowledge to create the tree model in Figure 2.11, let's consider the dataset in Table 2.3 to build a decision tree. The dataset has 6 samples, while each sample was collected empirically by observing the decision of a basketball team.

To build a decision tree model, we now face the first question, which is, in the root node, which variable should we use to define the first splitting rule. Possible splitting rules are {Weather = Rainy, Weather = Sunny, Dow = Saturday, Dow = Monday, Dow = Tuesday}. If we use the splitting rule, Weather = Sunny, it will result in the decision tree as shown in the left figure of Figure 2.12. If we use the splitting rule, Dow = Saturday, it will result in the decision tree as shown in the right figure of Figure 2.22. Which one should we use?

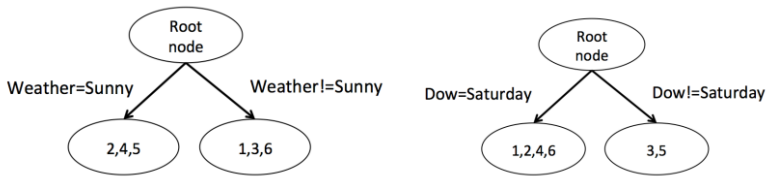

Figure 2.12: The decision tree models with Weather (left tree) or Dow (right tree) as the splitting variable in the root node

As we can see from Figure 2.12, once there is a set of splitting rule candidates, one of them needs to be selected at a node for splitting. To help us decide on which splitting rule is the best, the concept of **impurity** of data has been developed.

Impurity and information gain (IG): For classification problem (the outcome variable is categorical), the impurity of data points in a given node can be measured by **entropy**:

$$e = \sum_{i=1,..,K} -P_i log_2 P_i.$$

where K represents the number of classes and P_i is the proportion of data points in the node that belong to the class i. The entropy e is defined as 0 when the data points in the node all belong to one single class.

It is easy to see that, a node that has a large impurity is not ready to be a decision node yet. Thus, if we want to further split the node and create two more child nodes under it, we look for the best splitting rules that can minimize the impurity of the children nodes. This reduction of impurity can then be measured by **information gain** (**IG**), which is the difference of the entropy of the splitting node and the average entropy of the two children nodes weighted by their number of data points. The IG is defined as:

$$IG = e_s - \sum_{i=1,..,n} w_i * e_i.$$

Here, e_s is the entropy at the the splitting node, e_i is the entropy at the child node i, and w_i is the number of data points in the children node i divided by the number of data points at the splitting node[1].

To show how these concepts could be implemented, let's look at the decision tree model in the left figure of Figure 2.12. The entropy of the root node is calculated as

$$-\frac{4}{6}log_2\frac{4}{6} - \frac{2}{6}log_2\frac{2}{6} = 0.92.$$

The entropy of the left child node ("Weather = Sunny") is

$$-\frac{2}{3}log_2\frac{2}{3} - \frac{1}{3}log_2\frac{1}{3} = 0.92.$$

The entropy of the right child node ("Weather != Sunny") is 0 since all the three data points (ID = 1,3,6) belong to the same class.

---

[1] For regression problems, the variance of the outcome variable can be used for measuring the impurity of a node, i.e.,

$$v = \sum_{n=1}^{N}(\bar{y} - y_n)^2,$$

where $y_{n=1,...,N}$ are the values of the outcome variable, and $\bar{y}$ is the average of the outcome variable at the node. And the information gain can be calculated similarly to the classification problem.

The information gain for the splitting rule "Weather = Sunny" is then

$$IG = 0.92 - \frac{3}{6} * 0.92 - \frac{3}{6} * 0 = 0.46.$$

For the decision tree in the right figure of Figure 2.12, the entropy of the left child node ("Dow = Saturday") is

$$-\frac{2}{4} log_2 \frac{2}{4} - \frac{2}{4} log_2 \frac{2}{4} = 1.$$

The entropy of the right child node ("Dow != Saturday") is 0 since the two data points (ID = 3,5) belong to the same class.

Thus, the information gain for the splitting rule "Dow = Saturday" is then

$$IG = 0.92 - \frac{3}{6} * 1 - \frac{3}{6} * 0 = 0.42.$$

As the information gain of "Weather = Sunny" is higher, the splitting rule "Weather = Sunny" is preferred over its competitor "Dow = Saturday".

Similarly, the information gain can be calculated for all other splitting rules. As "Weather = Sunny" has the maximum information gain, it is selected for splitting the root node. The left child node with data points (ID = 2,4,5) still has two classes, and can be further split by selecting the next best splitting rule. The right child node has only one class and becomes a decision node labeled with the decision "play = No".

Note that, in our example, we only have categorical variables, while splitting rules could be easily defined. For continuously variables, the values of a variable are firstly ordered, and then, the average of each pair of consecutive values is used as the splitting value.

Greedy recursive approach to build a tree: Based on the concept of impurity and IG, we could develop a greedy strategy that recursively split the data points until there is no further IG, e.g., when there is only one data point, or only one class in the node. Most tree-building methds use this approach, with difference in the definitions of the impurity and IG.

However, this will inevitably lead to a very large decision tree with many inner nodes and unstable decision nodes (i.e., since the decisions assigned to these nodes are inferred empirically based on very few data points). Thus,

such a decision tree can be sensitive to noisy data, leading to worse accuracy and interpretability.

Tree pruning: To mitigate this problem, the pre-pruning or post-pruning methods can be used to control the complexity of a decision trees. Pre-pruning stops growing a tree when a pre-defined criterion is met. One can define the maximum depth of a tree, or minimum number of data points at each node. As these approaches are based on prior knowledge, they may not necessary reflect the data characteristics. More data-dependent approaches can be used. For example, the minimum impurity gain threshold can be used to stop growing a tree when the impurity gain is below the threshold. Still, a small impurity gain at a node does not necessarily indicate equivalent small impurity gain from its children nodes. Therefore, pre-pruning can cause over-simplified and thus under-fitted tree models. In other words, it is too cautious.

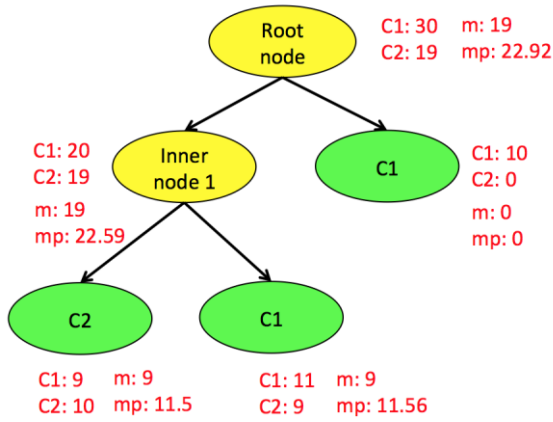

Figure 2.13: An example for tree pruning using pessimistic error

In contrast, post-pruning prunes a tree after it is fully-grown and has less risk of under-fit as a full-grown tree contains sufficient information captured from the data. Post-pruning starts from the bottom of the tree. If removing the sub-tree of an inner node does not increase the error, then

the sub-tree under the inner node can be pruned. Note that, the error here is not the error calculated based on the **training data** that is used to train the tree. This error calculated based on training data is called as **empirical error**. Rather, here, we should use the **generalization error**[1], that is, the error when applied to unseen data. Thus, in the famous decision tree algorithm, C4.5, the **pessimistic error estimation** approach is used.

We can derive the pessimistic error estimation using binomial approximation. Denote the empirical error rate on the training data as $\hat{e}$, which is only an estimation of the generalization error e. Since each data point can be either correctly or wrongly classified, we can view the probability of being correctly classified as a binomial distribution with probability e. With this insight, the normal distribution approximation can be applied here to derive the confidence interval of the generalization error e as:

$$\hat{e} - z_{\alpha/2}\sqrt{\frac{\hat{e}(1-\hat{e})}{n}} \leq e \leq \hat{e} + z_{\alpha/2}\sqrt{\frac{\hat{e}(1-\hat{e})}{n}}.$$

The upper bound of the interval, $\hat{e} + z_{\alpha/2}\sqrt{\frac{\hat{e}(1-\hat{e})}{n}}$, is used as the estimate of e. As this is an upper bound, it is named as pessimistic error estimation. Note that, the estimate of the error depends on three values, α, which is often set to be 0.25 so that $z_{\alpha/2}$=1.15; $\hat{e}$, which is the training error; and n, which is the number of data points at the node. Therefore, the estimated error is larger with a smaller n, accounting for the sample size as well.

Considering the tree in Figure 2.13 as an example. Each decision node is labeled with a class (i.e., either C1 or C2). Besides each node, we also highlight the distribution of the two classes, the misclassified instances (m), and the misclassified instances using the pessimistic error estimation (mp).

[1] We will return to this issue with more delicate discussion in Chapter 5.

Consider the inner node 1. We can see that, if we prune the subtree below inner node 1, we will label it with class C1, as 20 of the included data points are labeled as C1 while 19 are labeled as C2. And we can get that the total misclassified instances m is 19. Thus, $\hat{e} = \frac{19}{39} = 0.4871$. For the pessimistic error estimation, we can get that

$$\hat{e} + z_{\alpha/2}\sqrt{\frac{\hat{e}(1-\hat{e})}{n}} = 0.4871 + 1.15\sqrt{\frac{0.4871(1-0.4871)}{39}} = 0.579.$$

Thus, we can get that $mp = 0.579 \times 39 = 22.59$.

Now, let's see if the splitting of this inner node into its two child nodes improves the pessimistic error. It can be seen that, with this subtree of the two child nodes, the total misclassified instances m is 9+9=18. And by pessimistic error estimation, $mp = 11.5 + 11.56 = 23.06$ for the subtree. Therefore, based on the pessimistic error, we should prune the subtree.

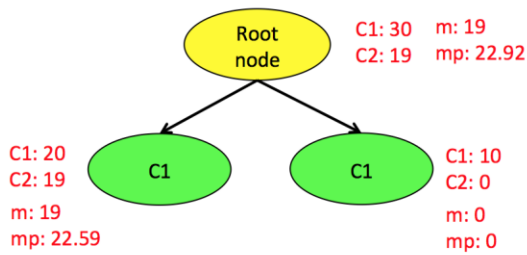

Figure 2.14: The pruned tree of Figure 2.13

Now the pruned tree is shown in Figure 2.14. Now consider whether to prune the children nodes of the root node. The pessimistic misclassified instances at the root node is 22.92, and the total pessimistic misclassified instances from its children nodes is 22.59+0=22.59. Pruning the children node would lead to increased error, and thus, the children nodes are kept and the final tree consists of three nodes.

III.3 R Lab

Now let's use the AD dataset for illustrating how the decision tree model can be used. Here, we use `DX_bl` as the outcome variable that is binary ("0" denotes for normal subjects while "1" denotes for diseased subjects), and use other variables (except `ID`, `TOTAL13` and `MMSCORE`) to predict `DX_bl`.

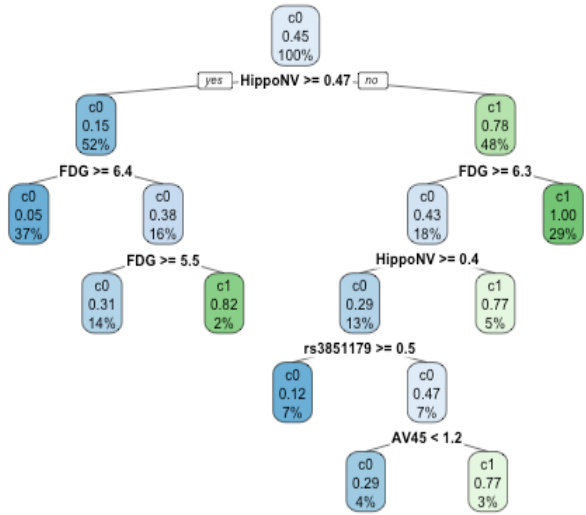

Figure 2.15: The unpruned decision tree model to predict `DX_bl` of the AD data

The R code in below loads the needed R packages and loads the data into the workspace.

```
library(rpart)
library(rpart.plot)
library(dplyr)
library(tidyr)
library(ggplot2)
library(partykit)
theme_set(theme_gray(base_size = 15))
```

```
path <- "../analytics/data/ AD_bl.csv"
data <- read.csv(path, header = TRUE)

target_indx <- which(colnames(data) == "DX_bl")
data[, target_indx] <- as.factor(paste0("c", data[, target_indx]))
rm_indx <- which(colnames(data) %in% c("ID", "TOTAL13", "MMSCORE
"))
data <- data[, -rm_indx]
```

The `rpart()` function in the R package "`rpart`" can be used to build the decision tree using the data and plot the decision tree in Figure 2.15.

```
tree <- rpart(DX_bl ~ ., data)
rpart.plot(tree, fallen.leaves = FALSE)
```

As an associated function with the `rpart` package, the importance score for each variable can be obtained from the tree. `HippoNV` has the largest importance score among all the variables.

```
print(tree$variable.importance)

##     HippoNV         FDG        AV45         AGE    PTGENDER
     e4_1
## 116.6665538  89.5608444  39.9595988  28.2195180  12.2040648
6.4708596
##   rs3851179    PTEDUCAT   rs3818361
##   4.2352941   1.1552265   0.8663915
```

The tree can be further pruned with the prune function whereas the parameter `cp` controls the model complexity. `cp` is the minimum relative error improved by splitting the node. A larger `cp` leads to a less-complex tree. First, let we try `cp = 0.05` which leads to Figure 2.16.

```
tree_0.05 <- prune(tree, cp = 0.05)
rpart.plot(tree_0.05, fallen.leaves = FALSE)
```

We can see that the tree is pruned. Then, we increase `cp` to `0.1` which leads to Figure 2.17.

```
tree_0.1 <- prune(tree, cp = 0.1)
rpart.plot(tree_0.1, fallen.leaves = FALSE)
```

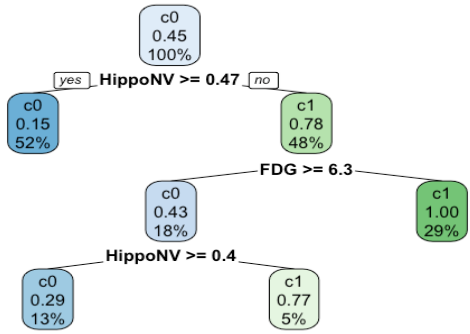

Figure 2.16: The pruned decision tree model to predict `DX_bl` of the AD data with `cp = 0.05`

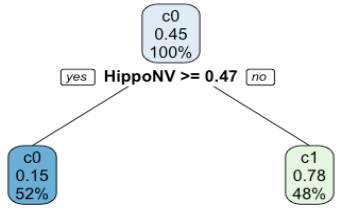

Figure 2.17: The pruned decision tree model to predict `DX_bl` of the AD data with `cp = 0.1`

We can see that, with `cp = 0.1`, the tree only has two nodes.

Now we have seen that the parameter `cp` could be used to control the model complexity in pruning. In practices, `cp` can be decided by minimizing the cross-validation error. The cross-validation will be introduced in detail in Chapter 5. Here, we take this opportunity to present it so we could get a sense of what it does. First, we split the data into halves, one half for training the tree model while another half for testing its accuracy. We then build a series of tree models using the training data with `cp` values ranging

from 0.2 to 0. For each built tree model, a training error and testing error can be calculated using the two datasets, respectively. For each tree, the number of decision nodes is recorded and used for measuring the complexity of the tree.

```
set.seed(1)
train.ix <- sample(nrow(data), floor(nrow(data)/2))
err.train.v <- NULL
err.test.v <- NULL
leaf.v <- NULL
for (i in seq(0.2, 0, by = -0.005)) {
    tree <- rpart(DX_bl ~ ., data = data[train.ix, ], cp = i)
    pred.train <- predict(tree, data[train.ix, ], type = "class")
    pred.test <- predict(tree, data[-train.ix, ], type = "class")
    current.err.train <- length(which(pred.train != data[train.ix,
 ]$DX_bl))/length(pred.train)
    current.err.test <- length(which(pred.test != data[-train.ix,
 ]$DX_bl))/length(pred.test)
    err.train.v <- c(err.train.v, current.err.train)
    err.test.v <- c(err.test.v, current.err.test)
    leaf.v <- c(leaf.v, length(which(tree$frame$var == "<leaf>")))
}
err.mat <- as.data.frame(cbind(train_err = err.train.v, test_err
= err.test.v,
    leaf_num = leaf.v))
err.mat$leaf_num <- as.factor(err.mat$leaf_num)
err.mat <- unique(err.mat)
err.mat <- err.mat %>% gather(type, error, train_err, test_err)

data.plot <- err.mat %>% mutate(type = factor(type))
ggplot(data.plot, aes(x = leaf_num, y = error, shape = type, colo
r = type)) +
    geom_line() + geom_point(size = 3)
```

The training errors and testing errors of the trees at different number of decision nodes are plotted in Figure 2.18. It can be seen that, as the complexity of trees increases, the training errors continue to decrease, while the testing errors first decrease but increase at some point. This indicates that, there is an optimal tree size that can be identified by testing error but will be missed by training error. This is actually the danger of aggressively pursuing models that can achieve too-good-to-be-true performances on training data, a commonly known phenomenon as **overfitting**.

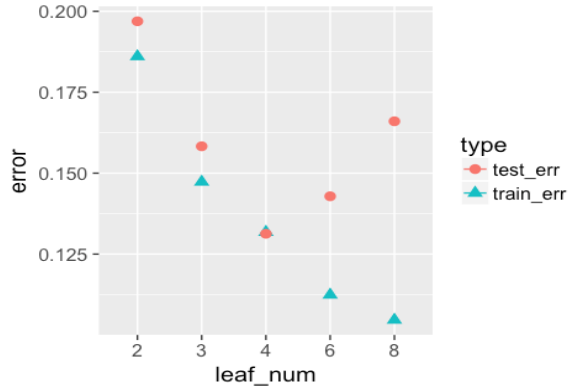

Figure 2.18: Training and testing errors versus complexity of the tree model (i.e., measured by the number of decision nodes in the tree)

III.4 Remarks

At each node of a decision tree, split is based on one single variable, and therefore, if the variable is continuous, the classification boundary for that split is perpendicular to the variable. And overall the classification boundary from a decision tree is always parallel or perpendicular to a continuous variable. This is illustrated in the following graph. The classification boundary consisting of splits by X_1 and X_2 is either parallel or perpendicular to one axis.

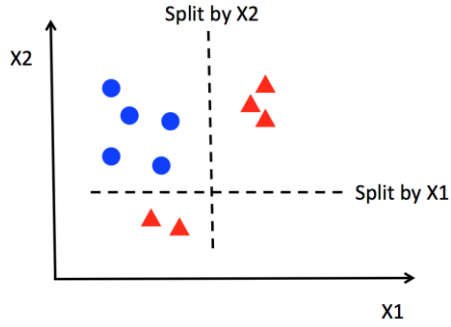

Figure 2.19: Decision boundary captured by tree models

This implies that, when applying a decision tree to a dataset with linear relationship between predictors and outcome variables, it may not be an optimal choice. In the following example, we simulate a data set and apply a decision tree and a logistics regression model (the counterpart of linear regression model for classification problem that will be introdueced in Chapter 3) to the data, respectively. The training data, and the predicted classes for each data point from the logistic regression and decision models are shown in Figures 2.20, 2.21 and 2.22, respectively. It can be seen the classification boundary from the logistics regression model is linear, while the one from the decision tree is parallel to the axis. This limitation makes a decision tree not be able to fully capture the linear relationship in the data.

```
ndata <- 2000
X1 <- runif(ndata, min = 0, max = 1)
X2 <- runif(ndata, min = 0, max = 1)
data <- data.frame(X1,X2)
data <- data %>% mutate( X12 = 0.5 * (X1 - X2), Y = ifelse(X12>=0,
1,0)  )
ix <- which( abs(data$X12) <= 0.05 )
data$Y[ix] <- ifelse(runif( length(ix)) < 0.5, 0, 1)
data <- data  %>% select(-X12) %>%  mutate( Y = as.factor(as.char
acter(Y) ))
ggplot(data,aes(x=X1,y=X2,color=Y))+geom_point()

linear_model <- glm(Y ~ ., family = binomial(link = "logit"), dat
a = data)
tree_model <- rpart( Y ~ ., data = data)
pred_linear <- predict(linear_model, data,type="response")
pred_tree <- predict(tree_model, data,type="prob")[,1]
data_pred <- data %>% mutate( pred_linear_class = ifelse( pred_li
near <0.5,0,1)  ) %>%
          mutate( pred_linear_class = as.factor(as.character(pred
_linear_class) )) %>%
          mutate( pred_tree_class = ifelse( pred_tree <0.5,0,1)  )
 %>%
          mutate( pred_tree_class = as.factor(as.character(pred_t
ree_class) ))
ggplot(data_pred,aes(x=X1,y=X2,color=pred_linear_class))+geom_poi
nt()

ggplot(data_pred,aes(x=X1,y=X2,color=pred_tree_class))+geom_point
()
```

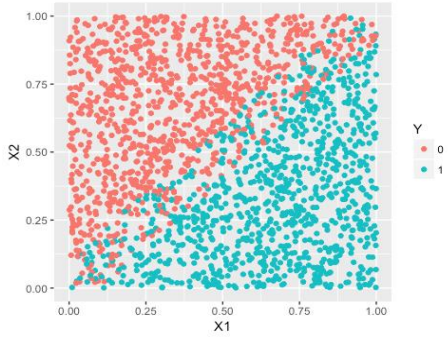

Figure 2.20: Scatterplot of the generated dataset

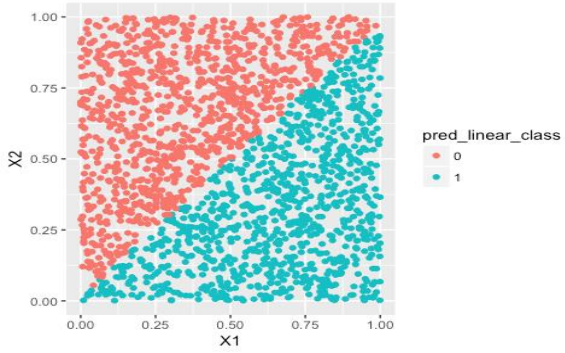

Figure 2.21: Decision boundary captured by logistic regression model

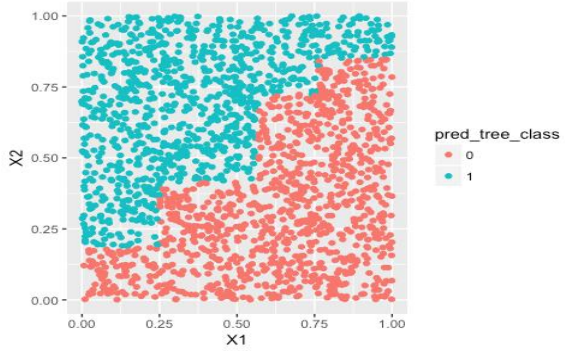

Figure 2.22: Decision boundary captured by the tree model

IV. Exercises

Data analysis

1. Repeat the analysis shown in the R lab of this chapter, but use `TOTAL13` as the outcome variable. Please build both the regression model and the decision tree model (for regression). Identify the final models you would select, evaluate the models, and compare the regression model with the tree model.
2. Find two datasets from the UCI data repository[1] or R datasets[2]. Conduct a detailed regression analysis for both datasets using both regression model and the tree model (for regression), e.g., for regression model, you may want to conduct model selection, model comparison, testing of the significance of the regression parameters, evaluation of the R-squared and significance of the model. Also comment on the application of your model on the context of the dataset you have selected.
3. Pick up any dataset you have used, and randomly split the data into two halves. Use one half to build the tree model and the regression model. Test the models' prediction performances on the second half. Report what you have found, adjust your way of model building, and suggest a strategy to find the model you consider as the best.

Derivation

4. Consider the case that, in building linear regression models, there is a concern that some data points may be more important (or more trustable). Thus, it is not uncommon to assign a weight to each data point. Denote the weight for the ith data point as w_i. We still want to estimate the regression parameters in the least squares framework. Follow the process of the derivation of the least

[1] http://archive.ics.uci.edu/ml/index.php
[2] https://vincentarelbundock.github.io/Rdatasets/datasets.html

squares estimator and propose your new estimator of the regression parameters.

5. Build a decision tree model based on the following dataset. Don't use R. Use your pen and paper, and show the process.

Table 2.4: dataset for building a decision tree

ID	X1	X2	Y
1	0.22	0.38	No
2	0.58	0.32	Yes
3	0.57	0.28	Yes
4	0.41	0.43	Yes
5	0.6	0.29	No
6	0.12	0.32	Yes
7	0.25	0.32	Yes
8	0.32	0.38	No

Programming

6. Write your own R script to implement the least squares estimation of a regression model. Compare the output from your script with the output from `lm()`.
7. As a continuation of 1, also write your own R script to derive the p-value of your regression parameters. Compare the output from your script with the output from `lm()`.
8. Write your own R script to build a decision tree using the greedy recursive process mentioned in this chapter, using the information gain. By your script, the tree growth process stops when a given depth of the tree is reached.