Introduction to Data Science Topic-6

- Instructor: Professor Henry Horng-Shing Lu,

  Institute of Statistics, National Yang Ming Chiao Tung University, Taiwan

  Email: henryhslu@nycu.edu.tw

- WWW: http://misg.stat.nctu.edu.tw/hslu/course/DataScience.htm
- Classroom: ED B27 (新竹市大學路1001號工程四館B27教室)
- References:

  M. A. Pathak, Beginning Data Science with R, 2014, Springer-Verlag.

  K.-T. Tsai, Machine Learning for Knowledge Discovery with R: Methodologies for Modeling, Inference, and Prediction, 2021, Chapman and Hall/CRC.
- Evaluation: Homework: 70%, Term Project: 30%
- Office hours: By appointment

Course Outline

10 Topics and 10 Homeworks:

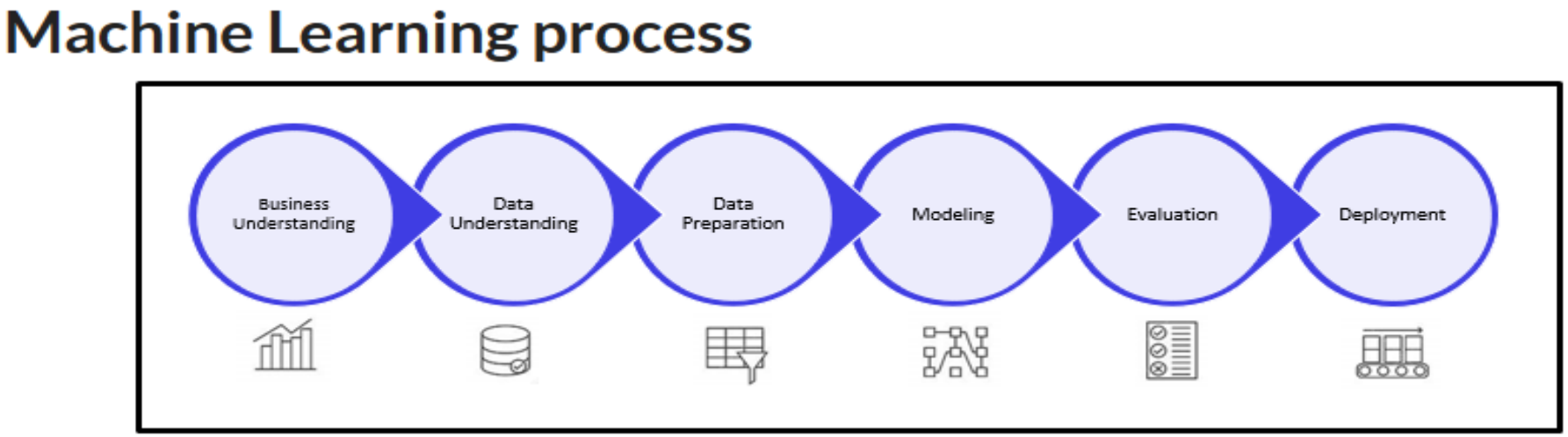
- **Introduction of Data Science**
- **Introduction of R and Python**
- **Cleaning Data into R and Python**
- **Data Visualization**
- **Exploratory Data Analysis**
- **Regression (Supervised Learning)**
- **Classification (Supervised Learning)**
- **Text Mining**
- **Clustering (Unsupervised Learning)**
- **Neural Network and Deep Learning**

Regression with R

References:
Ch. 6, M. A. Pathak, Beginning Data Science with R, 2014, Springer-Verlag.
https://www.kaggle.com/code/royshih23/topic6

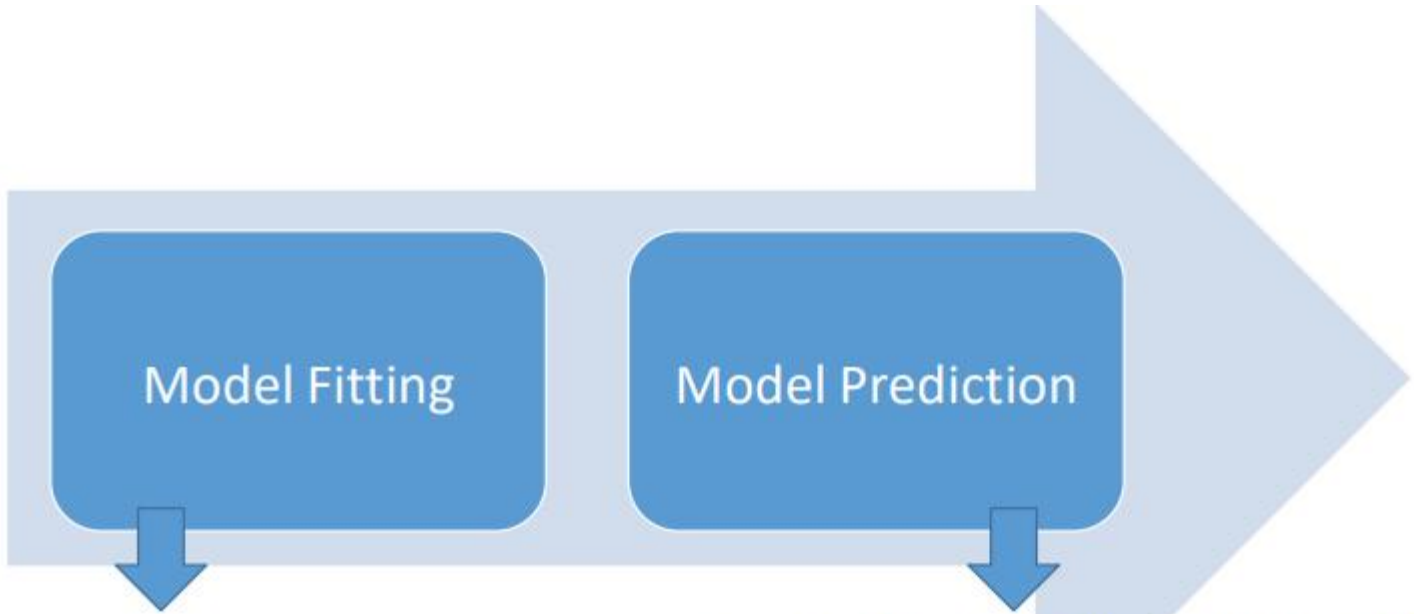
Before start

- ML/DL training and evaluation workflow



https://www.analyticsvidhya.com/blog/2021/04/rapid-fire-eda-process-using-python-for-ml-implementation/

Regression Techniques



For a given dataset with variables x1, . . . , xm and y we calculate the model parameters w that best meet certain criteria

Given a dataset with the predictor variables x1, . . . , xm and model parameters w1,…,wm, we calculate the value of the target variable y. The dataset could be the same one we use for model fitting or it could also be a new dataset.

Regression Techniques

Parametric Regression Models

- Simple Linear Regression
- Polynomial regression
- Multivariate Linear Regression
- Logarithmic Regressions

Nonparametric Regression Models

- Locally Weighted Regression
- Kernel Regression
- Regression Trees

What is Linear Regression?

- Linear regression is a very simple approach to supervised learning.
- A useful tool for predicting a quantitative response
- Linear regression assumes that the dependence on is linear

Simple Linear Regression

- We assume a model

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

- Given some estimates and for the model coefficients, we predict future dependent variables using

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

How to estimate the parameters of simple linear regression?

- Let be the prediction based on the value of X.

  represent the -ith residual

- Define the *residual sum of sauares* (RSS) as

$$RSS = e_1^2 + e_2^2 + \ldots + e_n^2$$

$$= (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \ldots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2$$

- The least squares approach chooses and minimizes the RSS. The minimizing values can be shown to be

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}, \text{ where } \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i, \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

KS

What is Simple Linear Regression?

Simple Linear Regression is a basic statistical method used to model the relationship between a single independent variable and a dependent variable. It's a fundamental technique in statistics and data analysis, particularly when you want to understand how changes in one variable are associated with changes in another.

Here are the key components and characteristics of Simple Linear Regression:

1. **Dependent Variable (Y)**: Simple Linear Regression focuses on a single dependent variable, often denoted as Y. This is the variable you are trying to predict or understand based on the values of the independent variable.
2. **Independent Variable (X)**: There's only one independent variable, typically denoted as X. This variable is used to explain or predict changes in the dependent variable.
3. **Linear Relationship**: Simple Linear Regression assumes that the relationship between the independent variable and the dependent variable is linear. In other words, it assumes that a change in X results in a proportional change in Y.

4. **Equation**: The relationship between X and Y is represented by the linear equation:
   $Y = \beta_0 + \beta_1 X + \varepsilon$
   - Y is the dependent variable.
   - X is the independent variable.
   - β_0 is the intercept (the value of Y when X is zero).
   - β_1 is the slope (the change in Y for a unit change in X).
   - ε represents the error term, which accounts for the variability in Y that cannot be explained by the linear relationship.
5. **Parameter Estimation**: The goal of Simple Linear Regression is to estimate the parameters (β_0 and β_1) that best fit the data. This is often done using a method such as the least squares method, which minimizes the sum of the squared differences between the observed Y values and the values predicted by the linear model.
6. **Interpretation**: The estimated values of β_0 and β_1 have specific interpretations. The intercept (β_0) represents the expected value of Y when X is zero, and the slope (β_1) represents the change in the expected value of Y for a one-unit change in X.
7. **Prediction**: Simple Linear Regression can be used for prediction. Given a value of X, you can use the linear equation to predict the corresponding value of Y.

Simple Linear Regression is widely used for tasks such as assessing the relationship between variables, making predictions, and understanding cause-and-effect relatio
serves as the foundation for more complex regression analysis, including multiple lin

Assessing the Accuracy of the Coefficient Estimates

- The standard error of an estimator reflects how it varies under repeated sampling. We have

$$SE(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}, SE(\hat{\beta}_0)^2 = \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \right], \sigma^2 = Var(\varepsilon)$$

- These standard errors can be used to compute confidence intervals. A 95% confidence interval is defined as a range of values such that with 95% probability, the range will contain the true unknown value of the parameter. It has the form

$$\hat{\beta}_1 \pm 2SE(\hat{\beta}_1)$$

Hypothesis testing

- Standard errors can also be used to perform hypothesis test on the coefficients. The most common hypothesis test involves testing the null hypothesis as follows:

  H_0 : There is no linear relationship between X and Y, $\beta_1 = 0$;

  H_1 : There is some relationship between X and Y, $\beta_1 \neq 0$.

- To test the null hypothesis, we compute a t-statistic, given by

$$t = \frac{\hat{\beta}_1 - 0}{SE(\hat{\beta}_1)}$$

- This will have a t-distribution with degrees of freedom

Assessing the Overall Accuracy of the Model

- We compute the *Residual Standard Error*

$$RSE = \sqrt{\frac{1}{n-2}RSS} = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- *R-square* or fraction of variance explained is

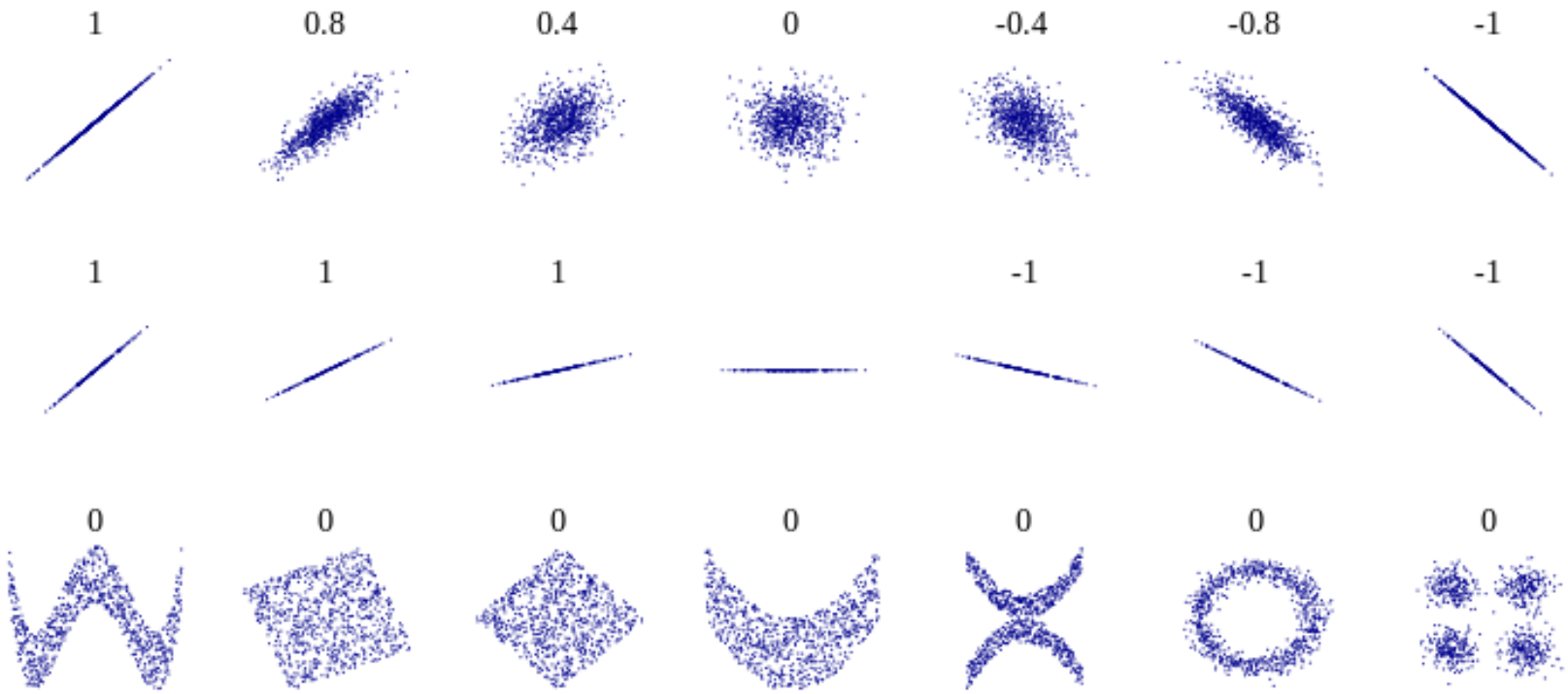
$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}, \text{ where } TSS = \sum_{i=1}^{n}(y_i - \bar{y})^2$$

TSS is the *total sum of squares*

- It can be shown that in this simple linear regression setting that where is the correlation between X and Y:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

correlation



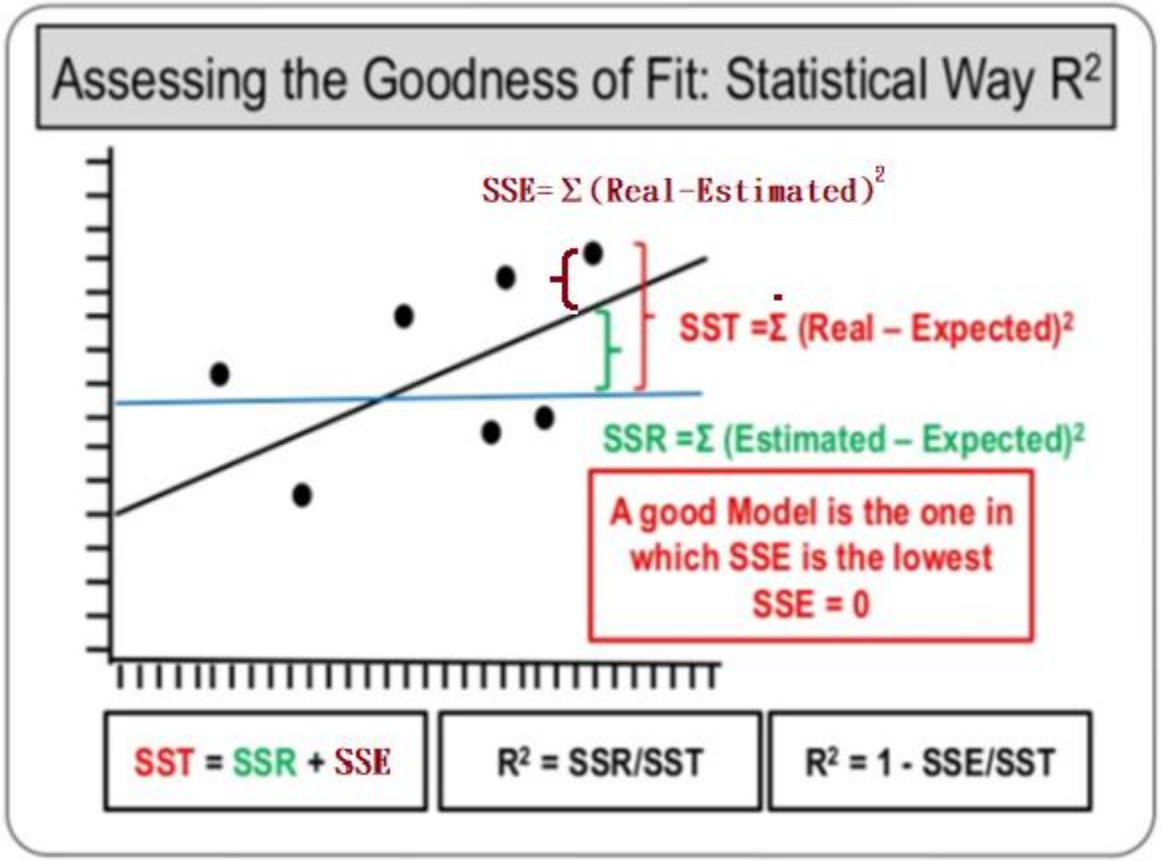
https://en.wikipedia.org/wiki/Correlation

Analysis methods

Features（Input、X)	Target（Output、Y)	Analysis method
continuous	continuous	Linear regression…
discrete	continuous	ANOVA test…
continuous	discrete	Logistic regression…
discrete	discrete	Chi square test…

ANOVA

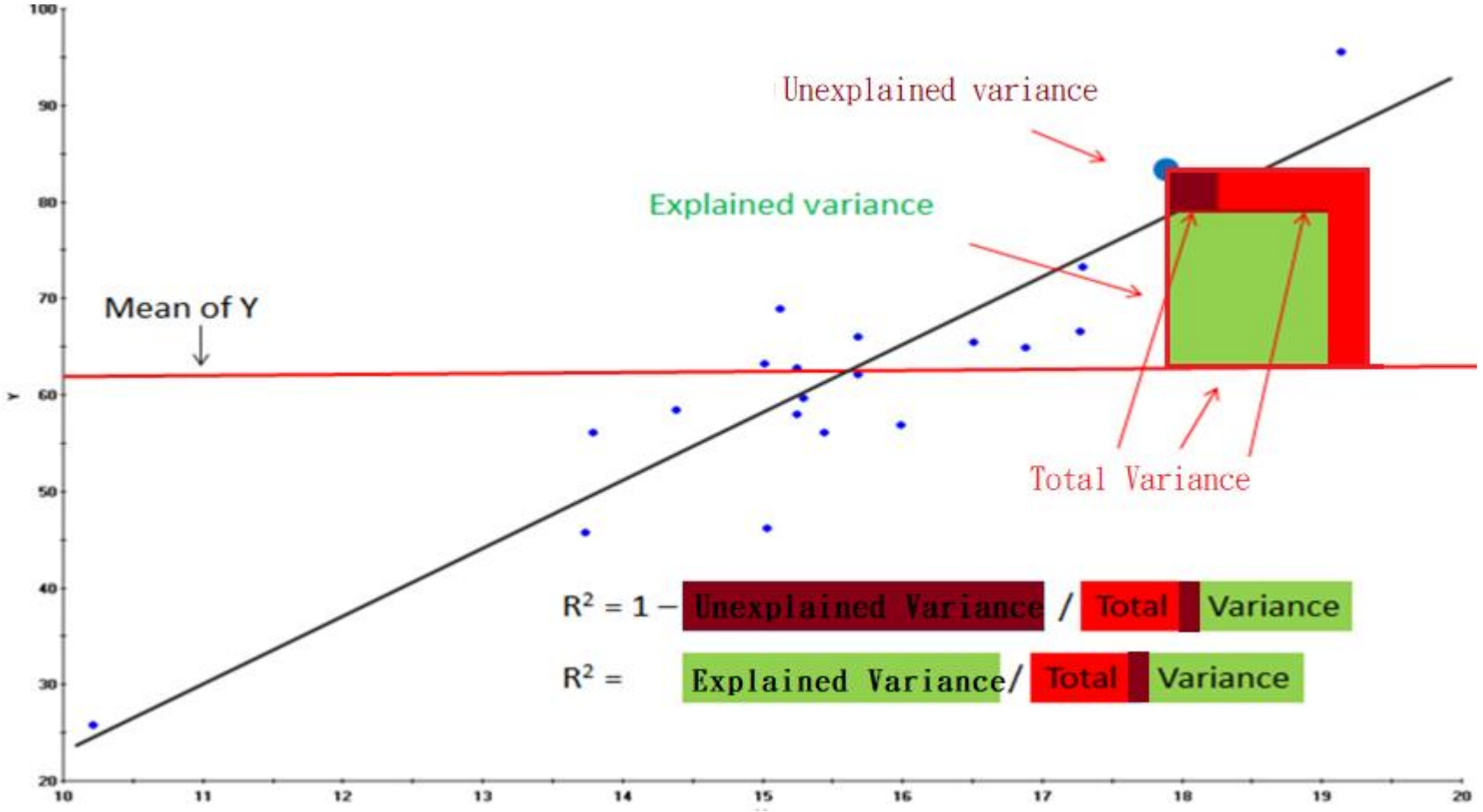
- Analysis Of Variance (ANOVA)



https://begin4learn.gitbooks.io/begin-to-learn-statistics/content/Statistics/ANOVA.html

ANOVA

- R-squared



https://begin4learn.gitbooks.io/begin-to-learn-statistics/content/Statistics/ANOVA.html

ANOVA

- ANOVA Tables

	Source of Variation	Degrees of Freedom	Mean of Squares	F-Ratio
variance of errors →	Within Groups	$n-k$	$MS_w = \frac{\sum\sum (y_{ij} - \bar{y}_j)^2}{n-k}$	MS_b / MS_w
variance of factors →	Between Groups	$k-1$	$MS_b = \frac{\sum n_j (\bar{y}_j - \bar{y})^2}{k-1}$	
total variance disregarding factors →	Total	$n-1$	$MS_{tot} = \frac{\sum\sum (y_{ij} - \bar{y})^2}{n-1}$	critical F value $F_{(k-1;n-k)}$

https://begin4learn.gitbooks.io/begin-to-learn-statistics/content/Statistics/ANOVA.html

Example for Simple Linear Regression

model=lm(price~horsepower,data=autos)
summary(model))
coef(model)
confint(model)#confidence interval of the estimated parameter

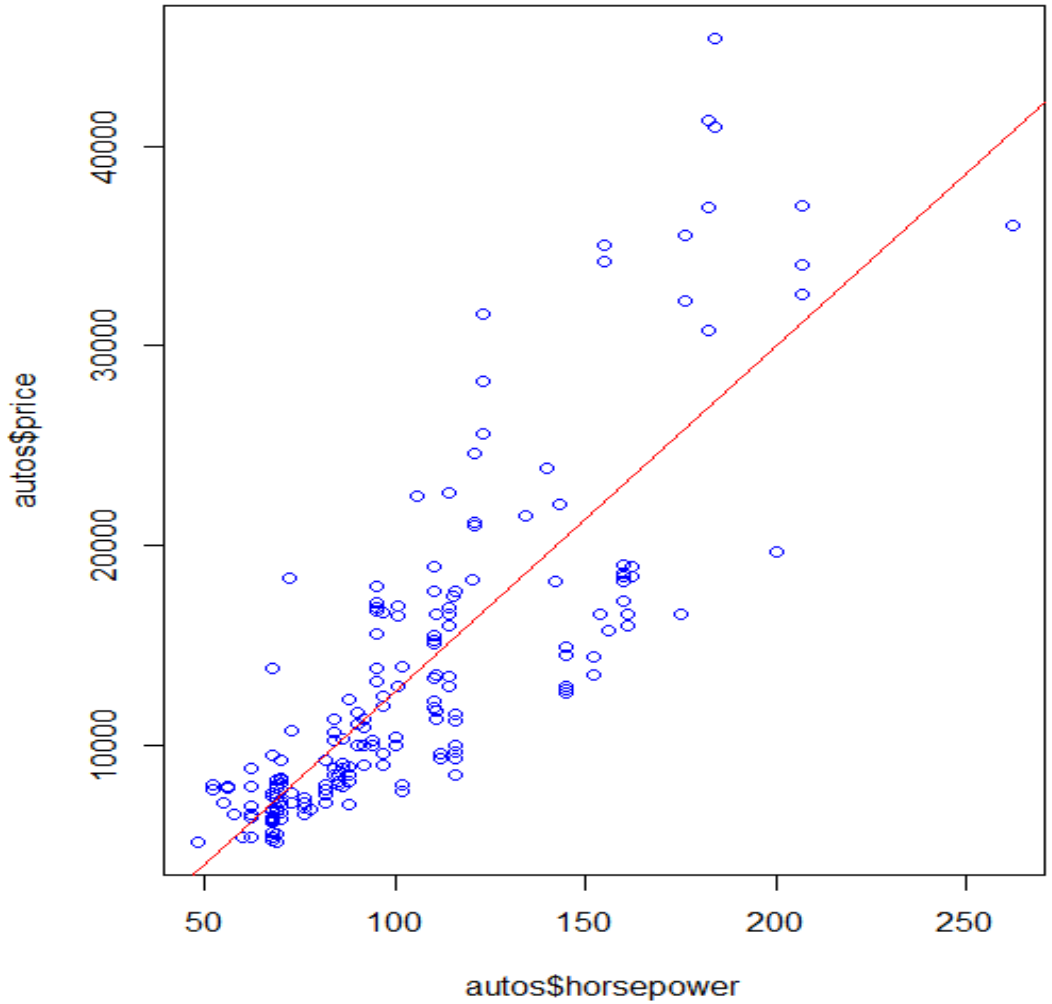
```
> model=lm(price~horsepower,data=autos)
> summary(model)

Call:
lm(formula = price ~ horsepower, data = autos)

Residuals:
     Min       1Q   Median       3Q      Max
-10296.1  -2243.5   -450.1   1794.7  18174.9

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -4630.70     990.58  -4.675 5.55e-06 ***
horsepower    173.13       8.99  19.259  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4728 on 191 degrees of freedom
Multiple R-squared:  0.6601,    Adjusted R-squared:  0.6583
F-statistic: 370.9 on 1 and 191 DF,  p-value: < 2.2e-16

> coef(model)
(Intercept)  horsepower
 -4630.7022    173.1292
```

Simple Linear Regression – Coefficient

```
> model = lm(price ~ horsepower)
> coef(model)
(Intercept) horsepower
-4630.7022   173.1292
```

β_0 β_1

```
>plot(horsepower,price)
> abline(model)
```



price = – 4630.7022 +173.1292 × horsepower

KS

what if the residuals are normal

If the residuals in a regression analysis are approximately normally distributed, it's a positive sign because it means that one of the key assumptions of linear regression has been met. This assumption, known as the normality of residuals, is important for the validity of statistical inference and hypothesis testing. Here's what it signifies:

1. **Valid Inference**: When residuals are normally distributed, it ensures that statistical tests and confidence intervals associated with the regression model are valid. In other words, p-values, confidence intervals, and hypothesis tests are reliable for making inferences about the relationships between the variables.
2. **Goodness of Fit**: Normally distributed residuals suggest that the linear regression model does a reasonably good job of capturing the underlying relationship between the independent and dependent variables.
3. **Interpretable Parameters**: Coefficients of the model (e.g., slopes and intercept) have meaningful interpretations when the residuals are normally distributed.

KS

what if the residuals are not normal

If the residuals in a regression analysis are not normally distributed, it indicates a violation of one of the key assumptions of linear regression. While non-normality doesn't necessarily invalidate the results, it can affect the reliability and interpretation of your model. Here's how to handle the situation when residuals are not normal:

1. **Assess the Extent of Non-Normality**:
   First, assess the degree of non-normality. Minor departures from normality, especially in large sample sizes, may not seriously affect the validity of your regression analysis. You can use graphical methods, such as a quantile-quantile (QQ) plot or a histogram of residuals, to visualize the distribution and identify the nature of non-normality.
2. **Consider the Central Limit Theorem**:
   Keep in mind that linear regression is somewhat robust to non-normality, thanks to the Central Limit Theorem. If the sample size is sufficiently large (usually around 30 or more observations), the distribution of the sample mean of residuals tends to approach normality, even if individual residuals are not normally distributed. In such cases, the results may still be reliable.
3. **Transform the Data**:
   If non-normality is observed, you can consider data transformation techniques. Common transformations include logarithmic, square root, or Box-Cox transformations. These transformations can often make the residuals more normally distributed.
4. **Use Robust Regression**:
   When non-normality is severe, and you believe that there are outliers or influential data points, consider using robust regression techniques. Robust regression methods are less sensitive to outliers and deviations from normality.

5. **Non-Parametric Methods**:
   If you cannot reasonably transform the data or if you have a small sample size, you might consider non-parametric regression methods, such as kernel regression or LOESS (locally weighted scatterplot smoothing), which are more flexible and don't rely on the normality assumption.
6. **Consider Generalized Linear Models (GLMs)**:
   In cases where the dependent variable is not normally distributed, you might need to use generalized linear models (GLMs) instead of simple linear regression. GLMs can handle different types of response distributions (e.g., Poisson, binomial) and link functions.
7. **Check for Violations of Other Assumptions**:
   Remember to assess other assumptions of linear regression, including linearity, independence, and homoscedasticity. Non-normality in conjunction with other violations can compound the problems.
8. **Seek Expert Advice**:
   If you're unsure about how to proceed or if non-normality is severe, consider consulting with a statistician or an expert in regression analysis.

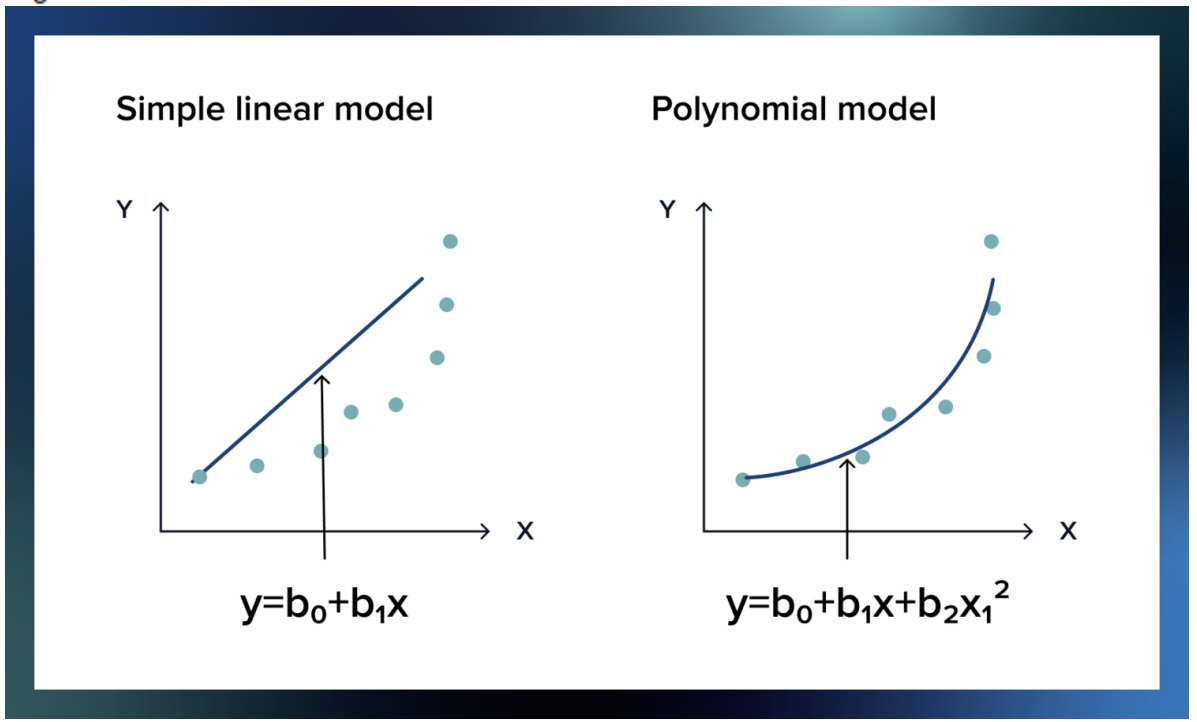
In summary, non-normality of residuals does not necessarily mean you need to discard your regression analysis. The impact of non-normality depends on its severity, the sample size, and the context of your research. There are techniques and methods available to address non-normality and improve the validity of your analysis.

Polynomial regression

In statistics, **polynomial regression** is a form of regression analysis in which the relationship between the independent variable *x* and the dependent variable *y* is modelled as an *n*th degree polynomial in *x*. Polynomial regression fits a nonlinear relationship between the value of *x* and the corresponding conditional mean of *y*, denoted E(*y* |*x*). Although *polynomial regression* fits a nonlinear model to the data, as a statistical estimation problem it is linear, in the sense that the regression function E(*y* | *x*) is linear in the unknown parameters that are estimated from the data. For this reason, polynomial regression is considered to be a special case of multiple linear regression.

In general, we can model the expected value of *y* as an *n*th degree polynomial, yielding the general polynomial regression model

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n + \varepsilon.$$



https://en.wikipedia.org/wiki/Polynomial_regression
https://serokell.io/blog/polynomial-regression-analysis

RO

What is Polynomial Regression?

Polynomial regression is a type of regression analysis in which the relationship between the independent variable (X) and dependent variable (Y) is modeled as an nth degree polynomial. In other words, it is a method of fitting a curve to data points that may not have a linear relationship.

The general equation for a polynomial regression of degree n is:

Y = a0 + a1*X* + *a2*X^2 + ... + an*X^n

where Y is the dependent variable, X is the independent variable, and a0, a1, a2, ..., an are the coefficients that represent the intercept and slope of the curve.

Polynomial regression can be useful when a linear model is not adequate to capture the underlying relationship between the variables. It allows for a more flexible curve that can better fit the data. However, it is important to note that higher degree polynomials can result in overfitting, meaning the model fits the data too closely and does not generalize well to new data. Therefore, it is important to carefully select the degree of the polynomial and evaluate the model's performance on a separate validation dataset.

Demo of Polynomial Regression In R

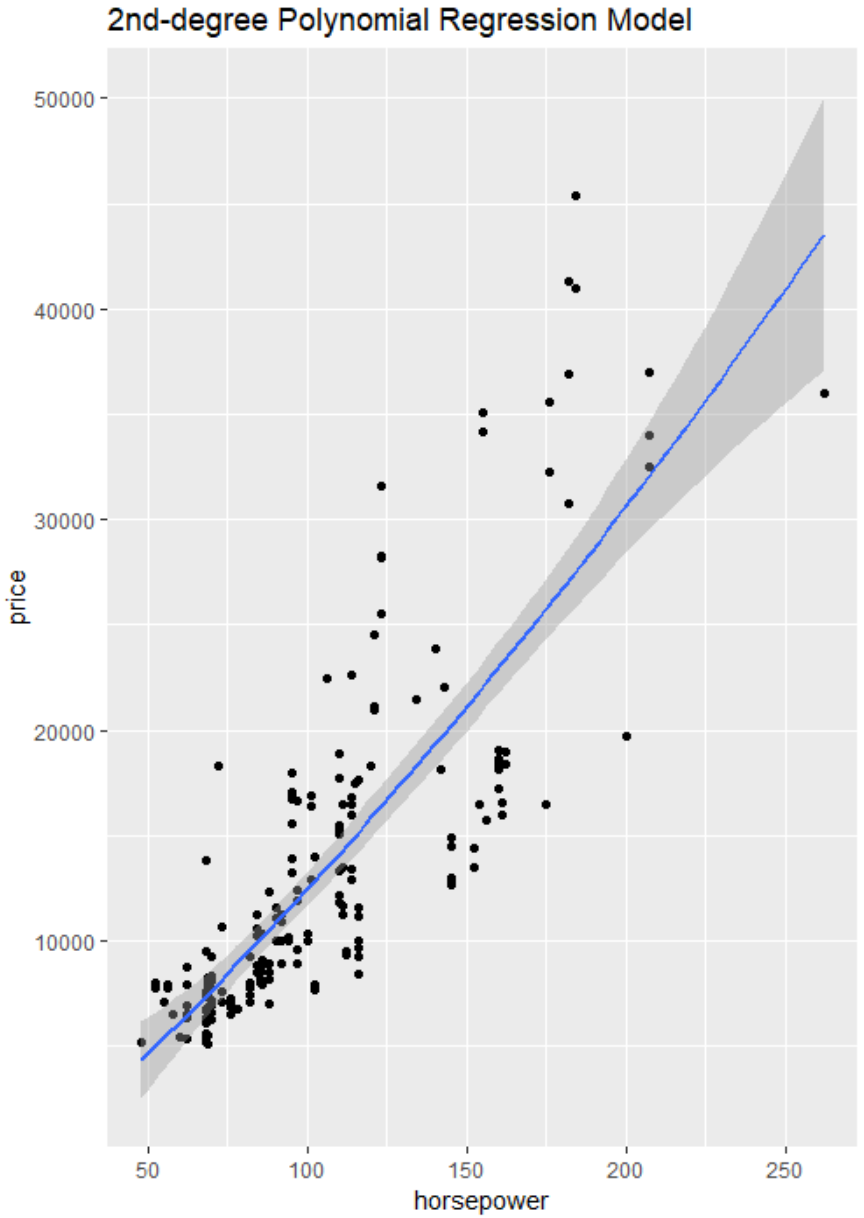
```r
```{r}
library("ggplot2")
#2nd-Degree Polynomial Regression:
# Fit a 2nd-degree polynomial regression model
model_2nd_degree <- lm(price ~ poly(horsepower, 2, raw = TRUE),data=autos)
model_2nd_degree
# Display the summary of the 2nd-degree polynomial regression model
summary(model_2nd_degree)
#predict(model_2nd_degree)
ggplot(autos, aes(horsepower, price) ) + geom_point() +
stat_smooth(method = lm, formula = y ~ poly(x, 2, raw = TRUE))
```
```

```
> # Display the summary of the 2nd-degree polynomial regression model
> summary(model_2nd_degree)

Call:
lm(formula = price ~ poly(horsepower, 2, raw = TRUE), data = autos)

Residuals:
     Min       1Q   Median       3Q      Max
-10920.6  -2169.5   -672.5   1773.8  17907.3

Coefficients:
                                Estimate Std. Error t value Pr(>|t|)
(Intercept)                    -2357.9241  2572.4712  -0.917  0.36051
poly(horsepower, 2, raw = TRUE)1   131.6898    44.2092   2.979  0.00327 **
poly(horsepower, 2, raw = TRUE)2     0.1660     0.1734   0.957  0.33960
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4730 on 190 degrees of freedom
Multiple R-squared:  0.6617,    Adjusted R-squared:  0.6582
F-statistic: 185.8 on 2 and 190 DF,  p-value: < 2.2e-16

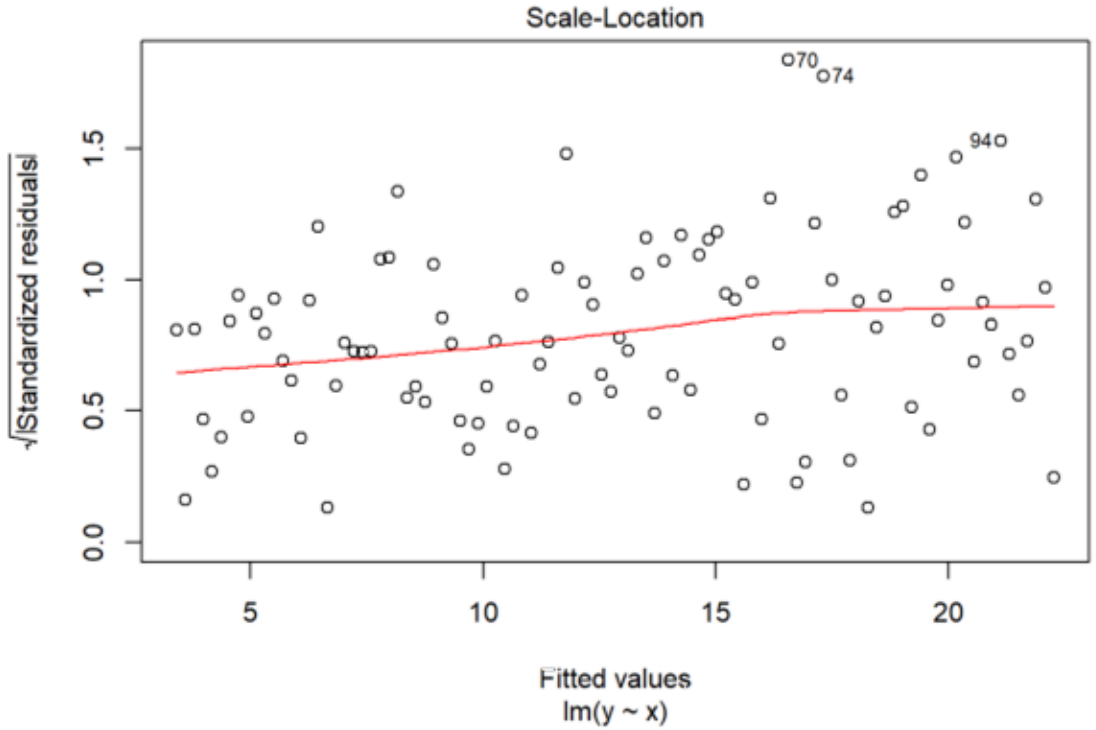
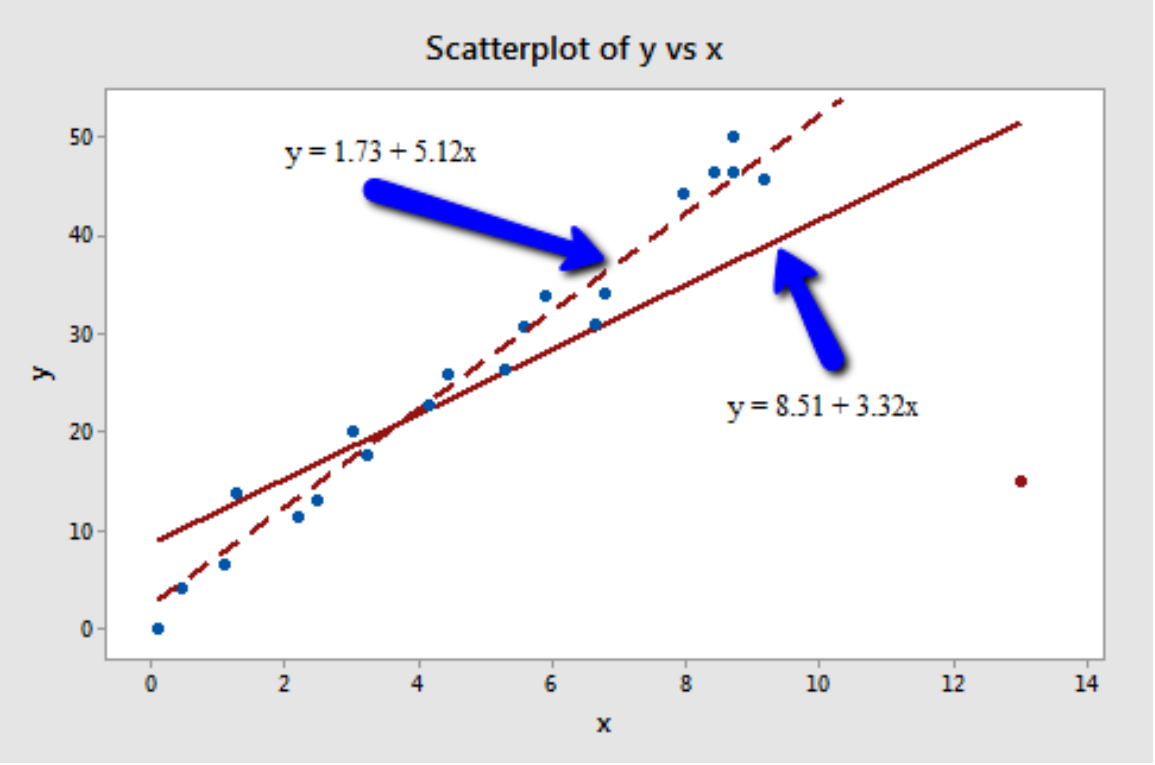
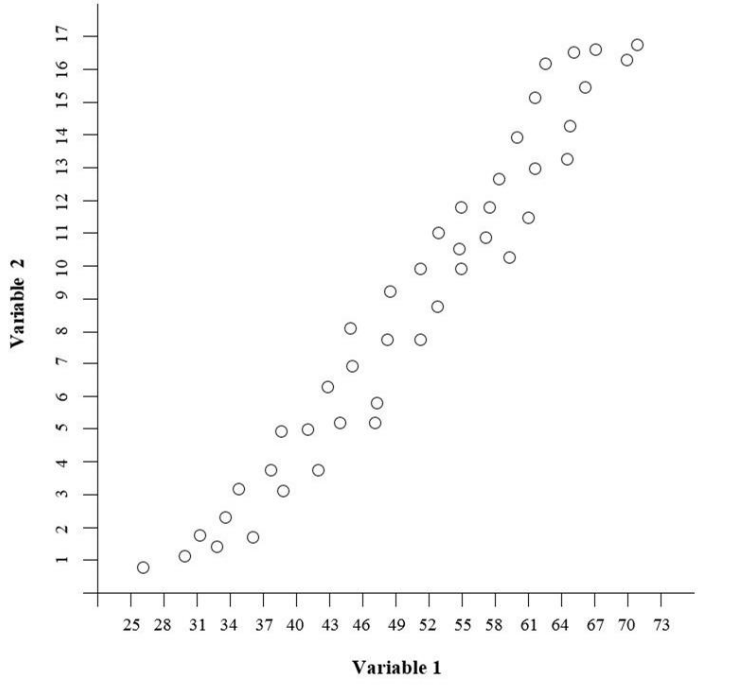
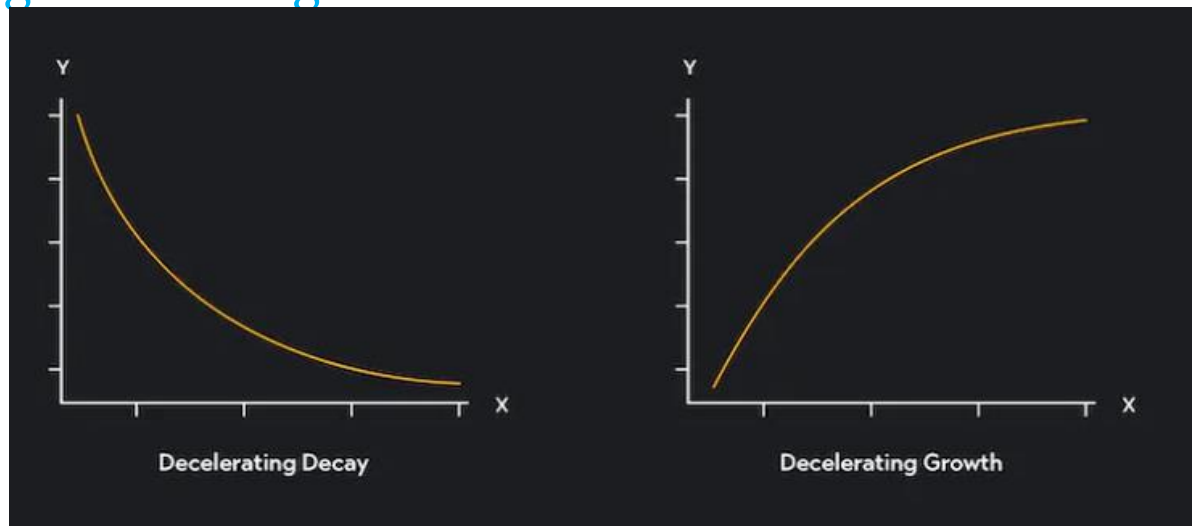
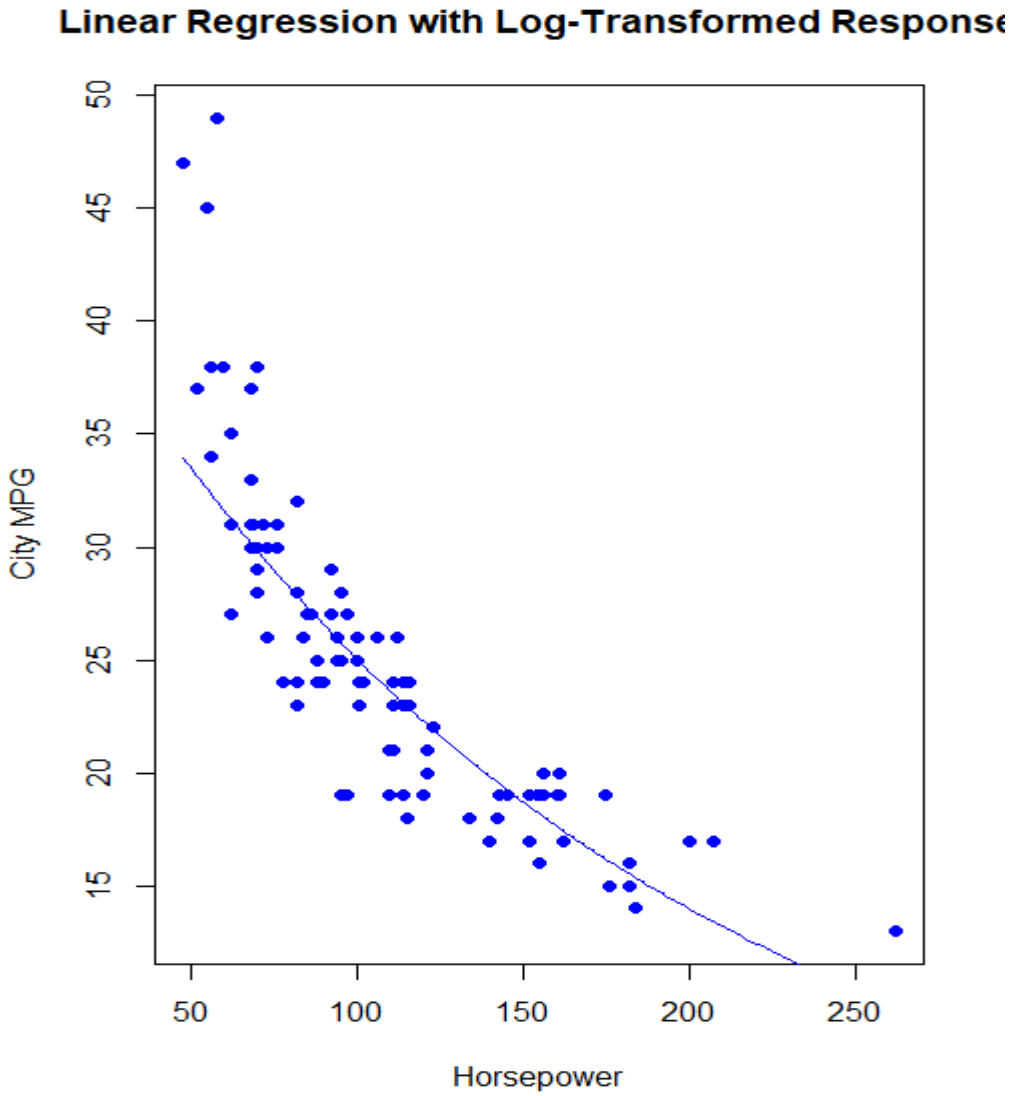
> #predict(model_2nd_degree)
> ggplot(autos, aes(horsepower, price) ) + geom_point() +
+ stat_smooth(method = lm, formula = y ~ poly(x, 2, raw = TRUE))
```



Multiple Linear Regression

- Model is

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \varepsilon$$

- We interpret as the average effect on Y of a one unit increase in, holding all other predictors fixed
- The ideal scenario is when the predictors are uncorrelated
- Correlations amongst predictors cause problems:

    -The variance of all coefficients tends to increase, sometimes dramatically

    -Interpretations become hazardous — when changes, everything else changes

- Claims of causality should be avoided for observational data.

Estimation and Prediction for Multiple Regression

- Given estimates, we can make predictions using this formula

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \ldots + \hat{\beta}_p x_p$$

- We estimate the values that minimize the sum of squared residuals

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$= \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \ldots - \hat{\beta}_p x_{ip})^2$$

- The values that minimize RSS are the *multiple least squares regression coefficient estimates*

Estimation and Prediction for Multiple Regression

- Let

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \qquad \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}, \qquad \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

- With this compact notation, the linear regression model can be written in the form $y = X\beta + \varepsilon$

Estimation and Prediction for Multiple Regression

In linear algebra terms, the least-squares parameter estimates β are the vectors that minimize

$$\sum_{i=1}^{n} \epsilon_i^2 = \epsilon'\epsilon = (\mathbf{y} - \mathbf{X}\beta)'(\mathbf{y} - \mathbf{X}\beta)$$

Any expression of the form $\mathbf{X}\beta$ is an element of a (at most) $(\mathbf{p}+1)$-dimensional hyperspace in $\mathbb{R}^n$ spanned by the $(\mathbf{p}+1)$ columns of X. Imagine the columns of $\mathbf{X}$ to be fixed, they are the data for a specific problem, and imagine β to be variable. We want to find the "best" β in the sense that the sum of squared residuals is minimized.

These vector normal equations are the same normal equations that one could obtain from taking derivatives. To solve the normal equations (i.e., to find the parameter estimates $\hat{\beta}$), multiply both sides with the inverse of $\mathbf{X}'\mathbf{X}$. Thus, the least-squares estimator of β is (in vector form)

$$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

Demo Example for Multiple Linear Regression

```
> model=lm(price ~ length + engine.size + horsepower + city.mpg,data=autos)
> summary(model)

Call:
lm(formula = price ~ length + engine.size + horsepower + city.mpg,
    data = autos)

Residuals:
    Min      1Q  Median      3Q     Max
-9697.0 -1745.7    24.9  1389.4 12904.6

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -28480.00    7114.51  -4.003 8.99e-05 ***
length         114.58      32.30   3.548 0.000491 ***
engine.size    115.32      12.92   8.922 4.06e-16 ***
horsepower      52.74      16.62   3.174 0.001756 **
city.mpg        61.51      83.05   0.741 0.459849
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3499 on 188 degrees of freedom
Multiple R-squared:  0.8168,    Adjusted R-squared:  0.8129
F-statistic: 209.5 on 4 and 188 DF,  p-value: < 2.2e-16

>
```

Important questions (1)

1. Is at least one of the predictors useful in predicting the response?
2. Do all the predictors help to explain Y, or is only a subset of the predictors useful?
3. How well does the model fit the data?
4. Given a set of predictor values, what response value should we predict, and how accurate is our prediction?

Important questions (2)

1. Is at least one of the predictors useful in predicting the response?

we can use the F-statistics

$$F = \frac{(TSS - RSS)/p}{RSS/(n-p-1)} \sim F_{p,n-p-1}$$

Important questions (3)

2. Do all the predictors help to explain Y, or is only a subset of the predictors useful?

We can use:

- Forward selection
- Backward selection
- Stepwise selection

Forward Selection

This procedure is based on the idea that no variables are in the model originally, but are added one at a time. The selection procedure is:

1. The first regressor selected to be entered into the model is the one with the highest correlation with the response. If the F statistic corresponding to the model containing this variable is significant (larger than some predetermined value, F_{in}), then that regressor is left in the model.
2. The second regressor examined is the one with the largest partial correlation with the response. If the *F*-statistic corresponding to the addition of this variable is significant, the regressor is retained.
3. This process continues until all regressors are examined.

Backward Elimination

This procedure is based on the idea that all variables are in the model originally, examined one at a time, and removed if not significant.

1. The partial F statistic is calculated for each variable *as if it were the last one added to the model*. The regressor with the smallest F statistic is examined first and will be removed if this value is less than some predetermined value F_{out}.
2. If this regressor is removed, then the model is refit with the remaining regressor variables and the partial F statistics are calculated again. The regressor with the smallest partial F statistic will be removed if that value is less than F_{out}.
3. The process continues until all regressors are examined.

Stepwise Regression

This procedure is a modification of forward selection.

1. The contribution of each regressor variable that is put into the model is reassessed by way of its partial F statistic.
2. A regressor that makes it into the model, may also be removed it if is found to be insignificant with the addition of other variables to the model. If the partial F-statistic is less than F_{out}, the variable will be removed.
3. Stepwise requires both a F_{in} value and F_{out} value.

3. How well does the model fit the data?

We can use:

- Mallow's Cp
- Akaike information criterion (AIC)
- Bayesian information criterion (BIC)
- adjusted R2

Adjusted R^2

- Say we are investigating a model with *p* terms,

$$R^2_{adj,p} = 1 - \left(\frac{n-1}{n-p}\right)(1 - R^2_p)$$

- This value will not necessarily increase as additional terms are introduced into the model. We want a model with the maximum adjusted R^2.

Residual Mean Square

- The MS_{res} for a subset regression model is

$$MS_{\mathrm{Re}\,s}(p) = \frac{SS_{\mathrm{Re}\,s}(p)}{n-p}$$

- $MS_{Res}(p)$ increases as p increases, in general. The increase in $MS_{Res}(p)$ occurs when the reduction in $SS_{Res}(p)$ from adding a regressor to the model is not sufficient to compensate for the loss of one degree of freedom. We want a model with a minimum $MS_{Res}(p)$.

Mallow's C_p Statistic

- This criterion is related to the MSE of the fitted value, that is

$$E\left[\hat{Y}_i - E(Y_i)\right]^2 = \left[E(Y_i) - E(\hat{Y}_i)\right]^2 + Var(\hat{Y}_i)$$

- where $\left[E(y_i) - E(\hat{y}_i)\right]^2$ is the *squared bias*. The total squared bias for a *p*-term model is

$$SS_B(p) = \sum_{i=1}^{n}\left[E(Y_i) - E(\hat{Y}_i)\right]^2$$

Mallow's C_p Statistic

- The standardized total squared error is

$$\Gamma_p = \frac{1}{\sigma^2}\left(\sum_{i=1}^{n}[E(Y_i) - E(\hat{Y}_i)]^2 + \sum_{i=1}^{n} Var(\hat{Y}_i)\right)$$
$$= \frac{SS_B(p)}{\sigma^2} + \frac{1}{\sigma^2}\sum_{i=1}^{n} Var(\hat{Y}_i)$$

- Making some appropriate substitutions, we can find the estimate of Γ_p, denoted C_p:

$$C_p = \frac{SS_{Res}(p)}{\hat{\sigma}^2} - n + 2p$$

Mallow's C_p Statistic

- It can be shown that if Bias = 0, the expected value of C_p is

$$E\left[C_p \mid Bias = 0\right] = \frac{(n-p)\hat{\sigma}^2}{\hat{\sigma}^2} - n + 2p = p$$

Mallow's C_p Statistic

Notes:

1. C_p is a measure of variance in the fitted values and (bias)^2. (Large bias can be a result of important variables being left out of the model).
2. C_p >> p, then significant bias.
3. Small C_p values are desirable.
4. Beware of negative values of C_p. These could result because the MSE for the full model overestimates the true σ^2.

AIC and BIC

- AIC based on maximizing the expected *entropy* of the model. Entropy is simply a measure of the expected information
- The key insight to the AIC is similar to R^2_{Adj} and Mallow's C_p . As we add regressors to the model, SS_{Res} cannot increase.
- The AIC and BIC criteria are gaining popularity. They are much more commonly used in the model selection procedures involving more complicated modeling situations than ordinary least square

Non-constant variance of error

Excessive non-constant variance can create technical difficulties with a multiple linear regression model. For example, if the residual variance increases with the fitted values, then prediction intervals will tend to be wider than they should be at low fitted values and narrower than they should be at high fitted values.



https://online.stat.psu.edu/stat462/node/186/

High leverage points

In statistics and in particular in regression analysis, leverage is a measure of how far away the independent variable values of an observation are from those of the other observations. High-leverage points, if any, are outliers with respect to the independent variables.



https://en.wikipedia.org/wiki/Leverage_(statistics)
https://online.stat.psu.edu/stat462/node/170/

Collinearity

When predictor variables in the same regression model are correlated, they cannot independently predict the value of the dependent variable. In other words, they explain some of the same variance in the dependent variable, which in turn reduces their statistical significance.



https://www.britannica.com/topic/collinearity-statistics

KS

Collinearity

Collinearity, also known as multicollinearity, is a common issue in regression analysis, where two or more independent variables in a regression model are highly correlated with each other. It can cause problems in regression analysis because it violates the assumptions of the model. Collinearity can manifest in the following ways:

1. **Perfect Collinearity:** This occurs when one independent variable can be perfectly predicted from one or more other independent variables in the model. In other words, there is a linear relationship between the predictors. This situation can lead to singular or ill-conditioned matrices, making it impossible to estimate unique coefficients.
2. **High Collinearity:** High collinearity exists when independent variables are strongly correlated, but not perfectly correlated. In this case, the correlation between predictors can make it difficult to isolate the individual effects of each variable on the dependent variable. It can lead to unstable coefficient estimates and make it challenging to interpret the significance of individual predictors.

Collinearity can be problematic for several reasons:

1. **Reduced Precision**: It can lead to imprecise coefficient estimates because the model has difficulty distinguishing the unique contributions of correlated variables.
2. **Interpretation Challenges**: Collinearity can make it challenging to interpret the relationships between independent variables and the dependent variable because the effects of individual variables may be confounded by their correlation with others.
3. **Unstable Coefficients**: Small changes in the data or the inclusion/exclusion of variables can result in significant changes in coefficient estimates.
4. **Hypothesis Testing**: Collinearity can affect the results of hypothesis tests, making some variables appear less significant than they actually are.

To deal with collinearity, consider the following strategies:

1. **Variable Selection**: Remove one of the correlated variables from the model. Choose the variable that is less theoretically important or less relevant to your research question.
2. **Combine Variables**: Create composite variables or indices that capture the essence of the correlated variables. This can reduce multicollinearity.
3. **Principal Component Analysis (PCA)**: Use PCA to transform correlated variables into orthogonal components, which can be used as independent variables in the regression model.
4. **Ridge Regression**: Ridge regression is a technique that adds a penalty term to the regression model to reduce the impact of multicollinearity.
5. **VIF (Variance Inflation Factor)**: Calculate the VIF for each predictor to quantify the degree of multicollinearity. A VIF greater than 10 is often considered a sign of problematic collinearity.
6. **Partial Correlation**: Use partial correlation to measure the relationship between the dependent variable and each independent variable while controlling for the effects of other variables.

Dealing with collinearity is essential for obtaining reliable and interpretable results in regression analysis. The specific strategy to address collinearity should be chosen based on the nature of the data and the research question.

Logarithmic Regressions

When Do We Use Logarithmic Regressions?

- Logarithmic regressions come in handy when your independent and dependent variables follow a nonlinear relationship matching the pattern of decelerating growth or decay. If your dependent variable increases or decreases rapidly at first as the independent variable decreases, but the rate of that increase or decrease gradually slows, you should consider using a logarithmic regression.



- Another case to use a logged variable is when you are dealing with a variable highly skewed (to the right) and follows a distribution called a log-normal distribution.

https://articles.outlier.org/logarithmic-regression

Types of Logarithmic Regressions

Three types of logarithmic regressions exist. In each type, you take the natural log, ln(x), of one or more of the variables in the regression equation.

1. Linear-log model:- In a linear-log model, you perform a log transformation on the independent variable.

$$Y_i = \beta_0 + \beta_1 \, ln(X_i) + \epsilon_i$$

2. Log-linear models:- In a log-linear model, you perform a log transformation on the dependent variable.

$$\ln(Y_i) = \beta_0 + \beta_1 \, X_i + \epsilon_i$$

3. Log-log models:- In a log-log model, you perform a log transformation on both the dependent and independent variables.

$$\ln(Y_i) = \beta_0 + \beta_1 \ln(X_i) + \epsilon_i$$

https://articles.outlier.org/logarithmic-regression

Demo Logarithmic Regressions

plot(autos$horsepower, autos$city.mpg, main="Logarithmic Regressions")




```
**Log-linear models**
```{r}
model_log1 <-lm(log(city.mpg) ~ horsepower,data=autos)
summary(model_log1)
```
> summary(model_log1)

Call:
lm(formula = log(city.mpg) ~ horsepower, data = autos)

Residuals:
     Min   1Q   Median   3Q   Max
-0.30581 -0.06380 0.00404 0.04964 0.42585

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.8041272 0.0245851 154.73   <2e-16 ***
horsepower -0.0058303 0.0002231 -26.13   <2e-16 ***

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1174 on 191 degrees of freedom
Multiple R-squared: 0.7814,   Adjusted R-squared: 0.7803
F-statistic: 682.9 on 1 and 191 DF, p-value: < 2.2e-16
```

Nonparametric Regression Models

- Nonparametric regression is a category of regression analysis in which the predictor does not take a predetermined form but is constructed according to information derived from the data. That is, no parametric form is assumed for the relationship between predictors and dependent variables.
- Nonparametric regression requires larger sample sizes than regression based on parametric models because the data must supply the model structure as well as the model estimates.

A non-exhaustive list of non-parametric models for regression.

- Nearest neighbors
- Regression trees
- kernel regression
- local regression
- multivariate adaptive regression splines
- smoothing splines
- neural networks

https://en.wikipedia.org/wiki/Nonparametric_regression

KS

What is Nonparametric Regression Models

Nonparametric regression models are a class of statistical models used for estimating the relationship between variables without making specific assumptions about the functional form of the relationship. Unlike parametric regression models (e.g., linear regression) that assume a fixed, predefined functional form for the relationship, nonparametric models are more flexible and data-driven, allowing the relationship to be estimated directly from the data. Here are the key characteristics of nonparametric regression models:

1. **No Fixed Functional Form**: Nonparametric regression models do not assume a specific mathematical equation (e.g., a linear, quadratic, or exponential relationship) between the independent and dependent variables. Instead, they allow the data to determine the shape of the relationship.
2. **Local Estimation**: Nonparametric models often use local estimation techniques, which means that they estimate the relationship within a local region of the data. The estimated function can vary across different parts of the data, adapting to the data's characteristics.
3. **Smoothing Techniques**: Smoothing techniques, such as kernel smoothing, spline smoothing, and loess (locally weighted scatterplot smoothing), are commonly used in nonparametric regression to create a smooth curve or surface that fits the data points.
4. **Data-Driven**: Nonparametric models are data-driven, meaning that the relationship between variables is estimated from the data itself. There is no need to make strong a priori assumptions about the data distribution or the shape of the relationship.

5. **Flexibility**: Nonparametric models are highly flexible and can capture complex, nonlinear relationships that may not be well-suited for parametric models. They are especially useful when the relationship between variables is not well-understood.
6. **Interpolation and Extrapolation**: Nonparametric models can handle both interpolation (predicting values within the observed range of the data) and extrapolation (predicting values outside the observed range) of the dependent variable.
7. **Regression Techniques**: Common nonparametric regression techniques include kernel regression, kernel density estimation, local regression, and smoothing splines.

Applications of nonparametric regression models include environmental modeling, finance, epidemiology, and any field where relationships between variables are complex and not easily captured by simple parametric models. Nonparametric regression can be particularly useful when dealing with noisy data, outliers, or situations where the relationship between variables may change over time or across different parts of the data.

In R, you can perform nonparametric regression using various functions and packages, including the `` `loess()` `` function for locally weighted scatterplot smoothing and the `` `ksmooth()` `` function for kernel smoothing, among others. These functions allow you to create smooth, data-driven estimates of the relationships between variables without making strict parametric assumptions.

Locally Weighted Regression(LWR)

- LOESS and LOWESS (locally weighted scatterplot smoothing) are two strongly related non-parametric regression methods that combine multiple regression models in a k-nearest-neighbor-based meta-model.

- "LOESS" is a later generalization of LOWESS; although it is not a true acronym, it may be understood as standing for "Local regression".

- It is a memory-based procedure because we need all the training data each time we wish to compute a prediction.

- The span plays a role like that of the tuning parameter λ in smoothing splines; it controls the flexibility of the non-linear fit.

RO

What is Local Regression?

Local regression, also known as LOESS (locally weighted scatterplot smoothing), is a non-parametric method for estimating a smooth curve through a set of data points. The method involves fitting a regression model to a subset of nearby data points, and then using the fitted model to estimate the value of the response variable at a specific point.

The basic idea of local regression is to use a weighted average of nearby data points to estimate the value of the response variable at a given point. The weights are determined by a kernel function that assigns higher weight to data points that are closer to the point of interest.

One popular kernel function used in local regression is the Gaussian kernel, which assigns higher weight to data points that are closer to the point of interest and lower weight to those that are farther away. The bandwidth of the kernel determines the width of the neighborhood of nearby data points that are used to fit the local regression model.

Local regression can be useful for modeling complex relationships between variables that may not be well captured by parametric models or splines. It can also handle data that has non-linear and non-monotonic patterns, and can be used for both univariate and multivariate regression.

However, local regression has some limitations. It can be sensitive to the choice of bandwidth and kernel function, and can be computationally intensive for large datasets. Careful selection of these parameters and validation of the model are important steps in using local regression effectively.

Overall, local regression is a powerful tool for modeling complex relationships between variables, and can provide a flexible and accurate representation of the underlying relationship between variables.

Locally Weighted Regression

- The smaller the value of the span *s*, the more *local* and wiggly will be our fit.
- A very large value of *s* will lead to a global fit to the data using all of the training observations.
- We can use cross-validation to choose *s* or specify it directly.
- Another choice to be made includes how to define the weighting function *K*, and whether to fit a linear, constant, or quadratic regression.

Locally Weighted Regression

Algorithm 7.1 *Local Regression At* $X = x_0$

1. Gather the fraction $s = k/n$ of training points whose x_i are closest to x_0.

2. Assign a weight $K_{i0} = K(x_i, x_0)$ to each point in this neighborhood, so that the point furthest from x_0 has weight zero, and the closest has the highest weight. All but these k nearest neighbors get weight zero.

3. Fit a *weighted least squares regression* of the y_i on the x_i using the aforementioned weights, by finding $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize

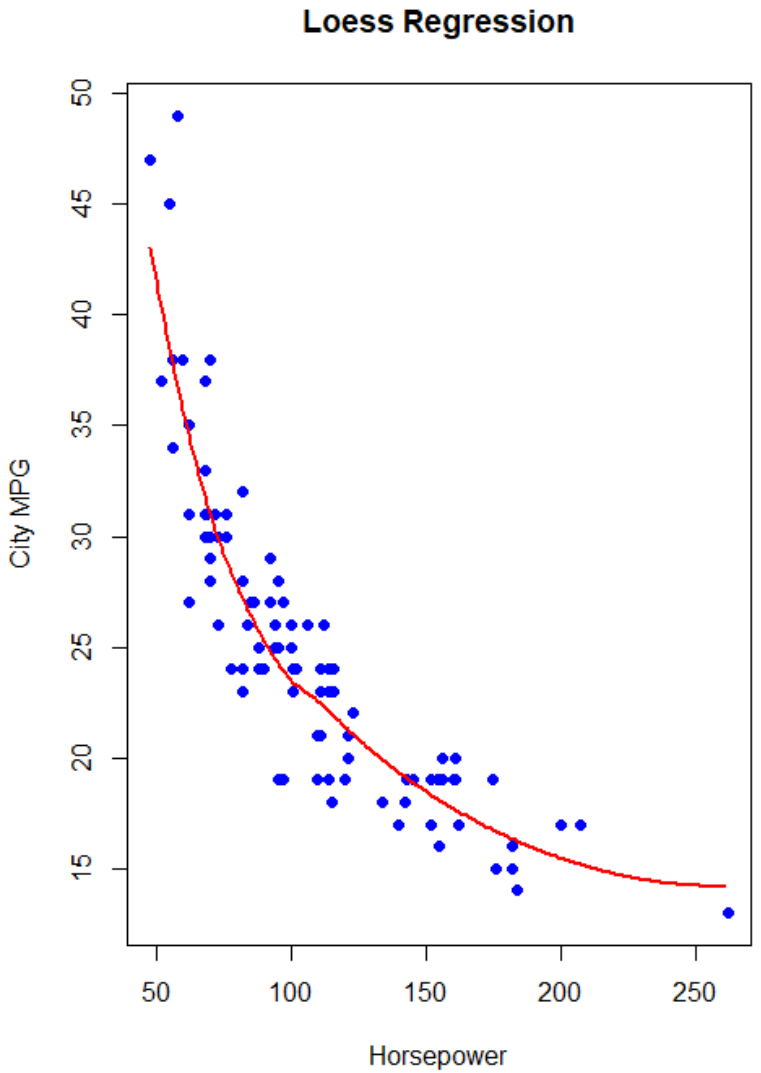
$$\sum_{i=1}^{n} K_{i0}(y_i - \beta_0 - \beta_1 x_i)^2. \tag{7.14}$$

4. The fitted value at x_0 is given by $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$.

Demo of Locally Weighted Regression

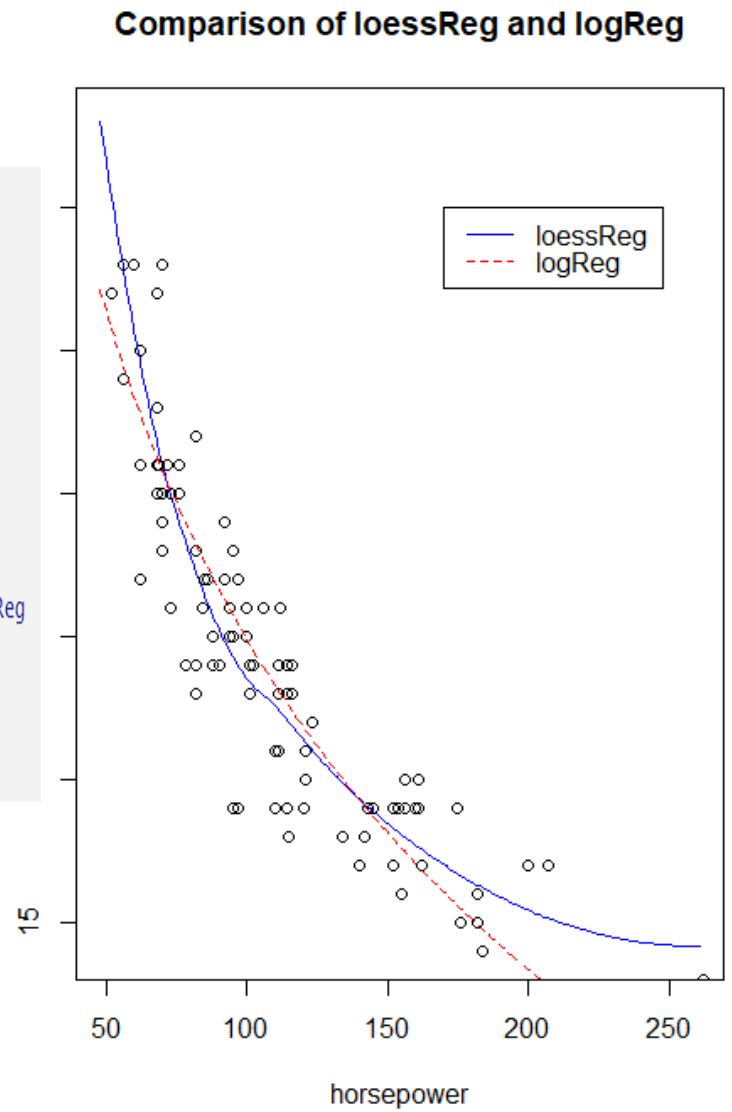
```r
# Fit a loess regression model
loess_model <- loess(city.mpg ~ horsepower,data=autos)
loess_model
# Generate a sequence of horsepower values
series = seq(48, 261,by=1)
# Predict city.mpg using the loess model
predictions_loess <- predict(loess_model, data.frame(horsepower = series))

# Create a scatterplot of the data
plot(autos$horsepower, autos$city.mpg, xlab = "Horsepower", ylab = "City MPG", pch = 19, col = "blue", main = "Loess Regression")

# Add the Loess regression curve to the plot
lines(series, predictions_loess, col = "red", lwd = 2)
```

```
> loess_model
Call:
loess(formula = city.mpg ~ horsepower, data = autos)

Number of Observations: 193
Equivalent Number of Parameters: 5.37
Residual Standard Error: 2.673
```



Comparison the log-linear regression model and loess model

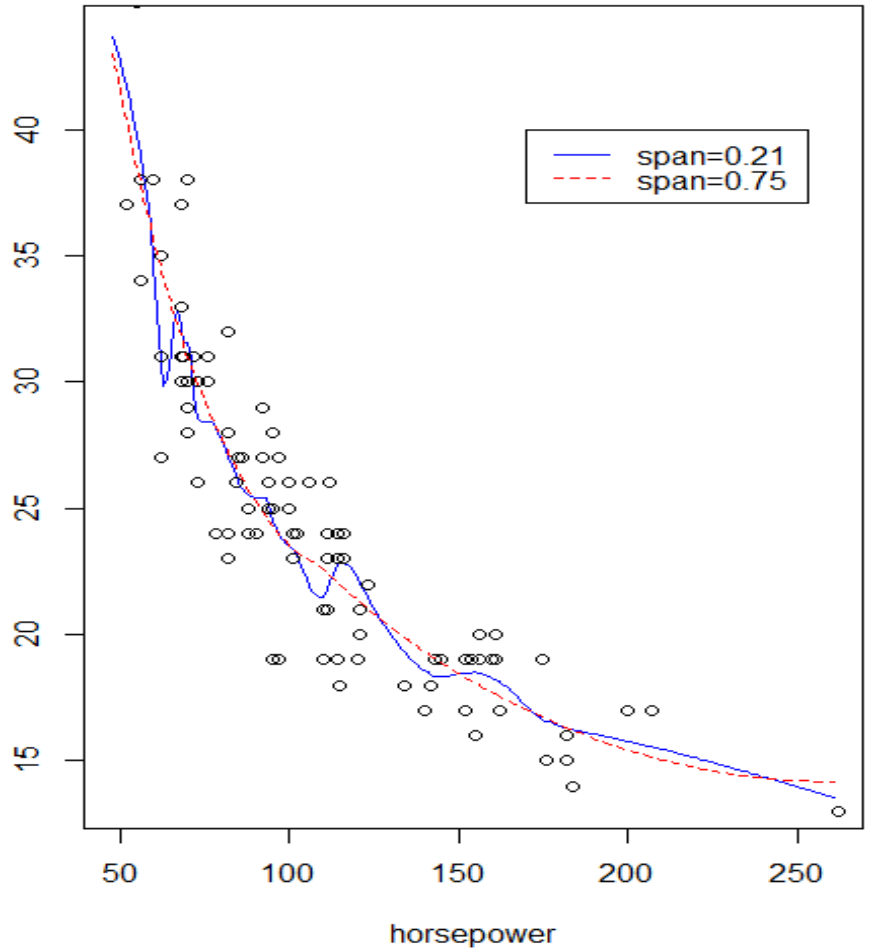
```r
# Fit a loess regression model
loessReg = loess(city.mpg ~ horsepower,data=autos)
# Fit a linear regression model with a log-transformed predictor
logReg = lm(city.mpg ~ log(horsepower),data=autos)
# Generate a sequence of horsepower values
series = seq(48, 261,by=1)
# Predict city.mpg using the loess model
predictions_loess <- predict(loessReg, data.frame(horsepower = series))
# Predict city.mpg using the linear model with a log-transformed predictor
predictions_logReg <- predict(logReg, data.frame(horsepower = series))
# Create a plot
plot(series,predictions_loess , col='blue', type ='l', lty=1, xlab="horsepower", ylab = "City MPG", pch = 19,main = "Comparison of loessReg and logReg")
points(autos$horsepower,autos$city.mpg)
lines(series,predictions_logReg , col='red', lty=2)
legend(170, 40,  legend = c("loessReg", "logReg"), col=c('blue', 'red'), lty=1:2)
```



Locally Weighted Regression

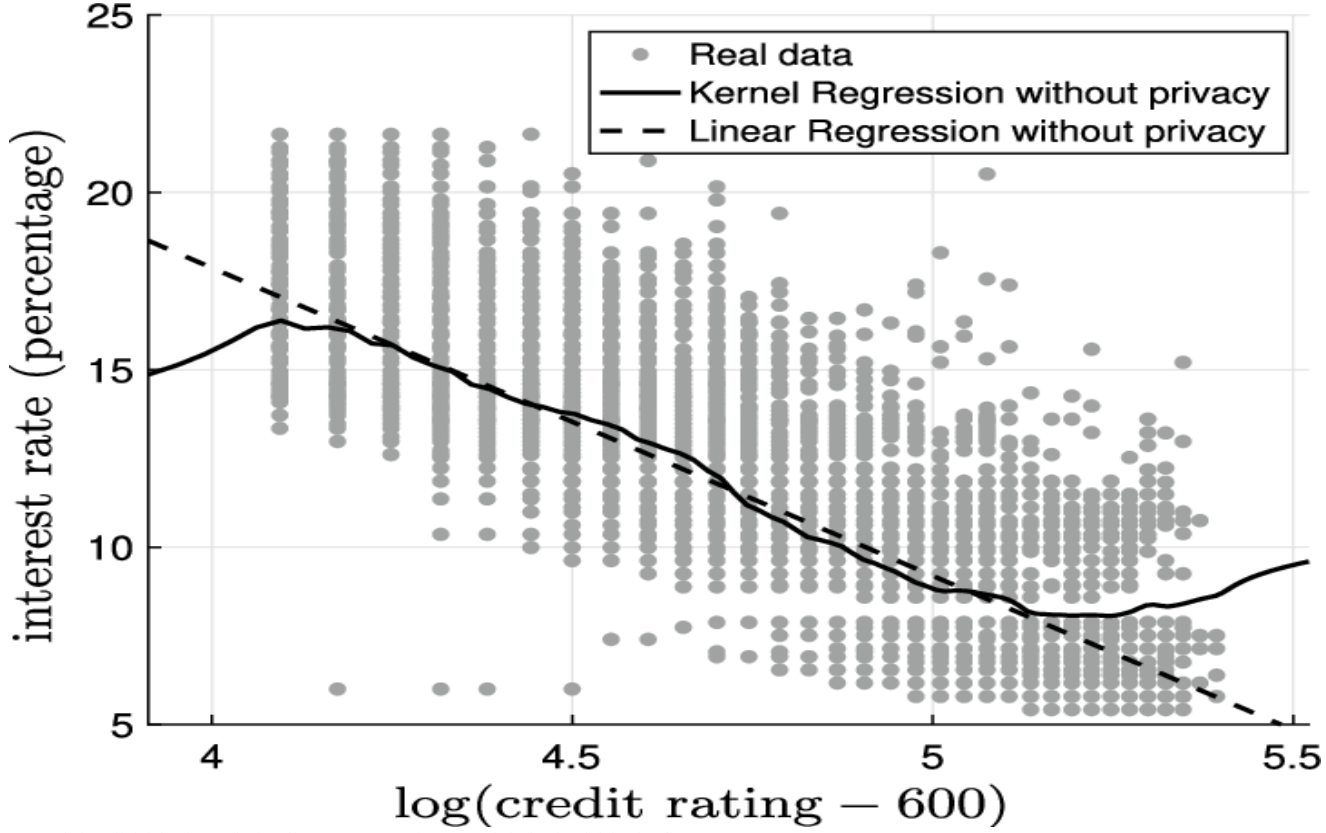
Change the span parameter

```{r}
loessReg1 = loess(city.mpg ~ horsepower, span=0.21,data=autos)
loessReg2 = loess(city.mpg ~ horsepower, span=0.75,data=autos)
series=seq(48, 261)
plot(series, predict(loessReg1, data.frame(horsepower=series)), col='blue', type ='l', lty=1, xlab="horsepower", ylab = "mpg")
points(autos$horsepower,autos$city.mpg)
lines(series, predict(loessReg2, data.frame(horsepower=series)), col='red', lty=2)
legend(170, 40, legend = c("span=0.21", "span=0.75"), col=c('blue', 'red'), lty=1:2)
```



Kernel Regression

- Kernel regression is another nonparametric approach where we compute the value of the predictor variable at each data point by taking a weighted average of the target variable at all data points.
- The weights are given by the kernel function, which we can think of as a measure of distance between two data points.
- The data points nearer to the candidate data point have high weight and those further away from the data point have low weight.



https://www.researchgate.net/figure/The-kernel-regression-model-solid-black-and-the-linear-regression-model-dashed-black_fig1_346693436

KS

what is Kernel Regression ?

Kernel regression, also known as kernel smoothing or kernel density estimation, is a non-parametric statistical method used to estimate a smooth curve or surface from a set of data points. Unlike parametric regression models that assume a specific functional form for the relationship between variables (e.g., linear regression), kernel regression is a flexible and data-driven technique that adapts to the underlying structure of the data.

Here are the key characteristics of kernel regression:

1. **Data-Driven Smoothing**: Kernel regression is particularly useful when the relationship between variables is nonlinear or when there is significant noise in the data. It estimates the relationship between variables without making strong parametric assumptions.
2. **Kernel Function**: The central concept in kernel regression is the kernel function. The kernel function assigns weights to data points based on their proximity to the point of interest. Common kernel functions include the Gaussian (normal) kernel, Epanechnikov kernel, and others. The choice of kernel function and bandwidth parameter is essential and affects the smoothness of the estimated curve.
3. **Local Estimation**: Kernel regression focuses on local estimation. It assigns higher weights to data points that are closer to the point being estimated and lower weights to more distant points. This allows the estimated curve to adapt to the data's characteristics, providing more detail in areas with dense data and smoothing out noise in sparser regions.

4. **Bandwidth Parameter**: The bandwidth parameter controls the width of the kernel function. A smaller bandwidth results in a more detailed, wiggly curve that follows the data closely, while a larger bandwidth leads to a smoother, less wiggly curve.
5. **Adaptive Smoothing**: Kernel regression is adaptive, meaning it provides more smoothing in regions where data points are sparse and less smoothing in regions with dense data.
6. **Applications**: Kernel regression is used in various fields, including statistics, economics, finance, and environmental science, for estimating probability density functions, non-parametric regression, and data visualization.

In R, you can perform kernel regression using functions like `` `ksmooth()` `` or `` `density()` ``. Kernel regression is a valuable tool for modeling relationships in data when parametric models may not be suitable, especially when dealing with complex, nonlinear relationships or noisy data. The choice of kernel function and bandwidth is crucial for achieving the desired level of smoothing.

KS

when to use Kernel Regression ?

Kernel regression, also known as kernel smoothing or kernel density estimation, is a valuable statistical technique that can be employed in various situations. Here are some scenarios when you might consider using kernel regression:

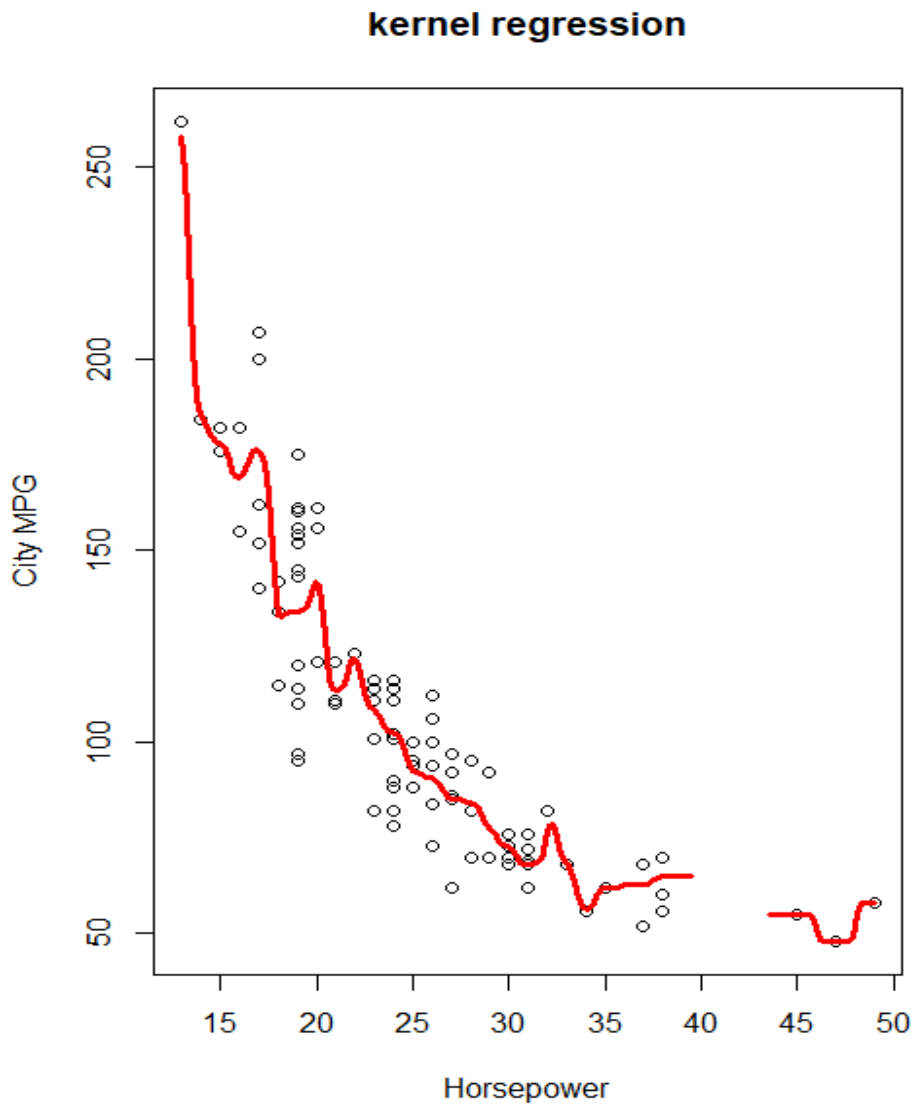
1. **Nonlinear Relationships**: When the relationship between your independent and dependent variables is nonlinear and cannot be adequately modeled using parametric regression models (e.g., linear regression). Kernel regression is highly flexible and can capture complex, nonlinear patterns in the data.
2. **Noisy Data**: In situations where the data is noisy, kernel regression can help smooth out the noise, providing a clearer and more interpretable representation of the underlying trend in the data.
3. **Density Estimation**: Kernel regression is commonly used for density estimation, particularly when you want to estimate the probability density function (PDF) of a continuous random variable. This is useful in fields like environmental science, finance, and image processing.
4. **Local Estimation**: When you are interested in understanding the local behavior or trends in your data. Kernel regression focuses on local estimation, adapting to the data's characteristics in different regions, and providing a more detailed picture of the data's structure.

5. **Data Exploration and Visualization**: Kernel regression is often used for data visualization. It can help you create smooth curves that highlight the central tendency of your data, making it easier to visualize and understand the data distribution.
6. **Small Sample Size**: In cases where you have a small sample size, kernel regression can be more robust than parametric methods that require strong assumptions about the underlying distribution.
7. **Model Comparison**: When you want to compare the performance of different models, including parametric and non-parametric models. Kernel regression can serve as a benchmark for understanding the data's characteristics.
8. **Regression Diagnostics**: Kernel regression can be useful for preliminary regression diagnostics. You can use it to visualize the data and identify potential issues like heteroscedasticity or nonlinearity.
9. **Data Preprocessing**: Kernel regression can also be used as a preprocessing step in data analysis. By smoothing the data, you can reduce the dimensionality and noise, making it easier to apply other analysis techniques.
10. **Geospatial Data**: Kernel regression is suitable for spatial data analysis. It can help estimate spatial patterns and trends, which is important in fields like geography and environmental science.

It's important to note that while kernel regression has many advantages, it also has limitations. The choice of kernel function and bandwidth parameter is crucial and can significantly impact the results. Additionally, kernel regression may not be appropriate for all data types, and the choice of regression method should be guided by the specific characteristics of your dataset and research goals.

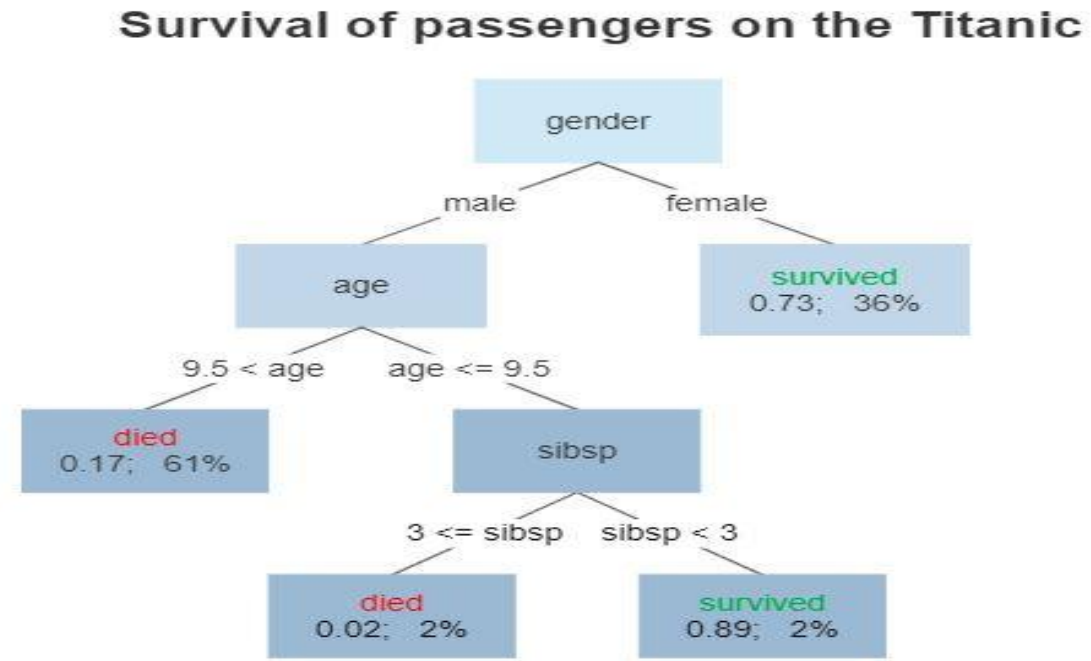
Kernel Regression – Kernel Functions

```r
# Perform kernel regression
ks_model <- ksmooth(autos$city.mpg, autos$horsepower, "normal", 1)
summary(ks_model)
# Create a scatterplot of the original data
plot(autos$city.mpg, autos$horsepower, xlab = 'Horsepower', ylab = 'City MPG', main="kernel regression")

# Overlay the kernel regression curve on the plot
lines(ks_model$x, ks_model$y, type = 'l', col = 'red', lwd = 3)
```



Regression Trees

- Decision trees are one of the most widely used models in all of machine learning and data mining.
- A tree is a data structure where we have a root node at the top and a set of nodes as its children. The child nodes can also have their own children, or be terminal nodes in which case they are called leaf nodes.
- A tree has a recursive structure as any of its node is the root of a subtree comprising the node's children.



https://en.wikipedia.org/wiki/Decision_tree_learning#/media/File:Decision_Tree.jpg

KS

what is Regression Trees?

Regression Trees are a type of decision tree used in predictive modeling and statistical analysis, primarily for solving regression problems. They are a machine learning and statistical modeling technique that is particularly useful when you want to predict a continuous numerical outcome, such as a price, temperature, or sales, based on input features or variables. Here are the key characteristics and concepts associated with Regression Trees:

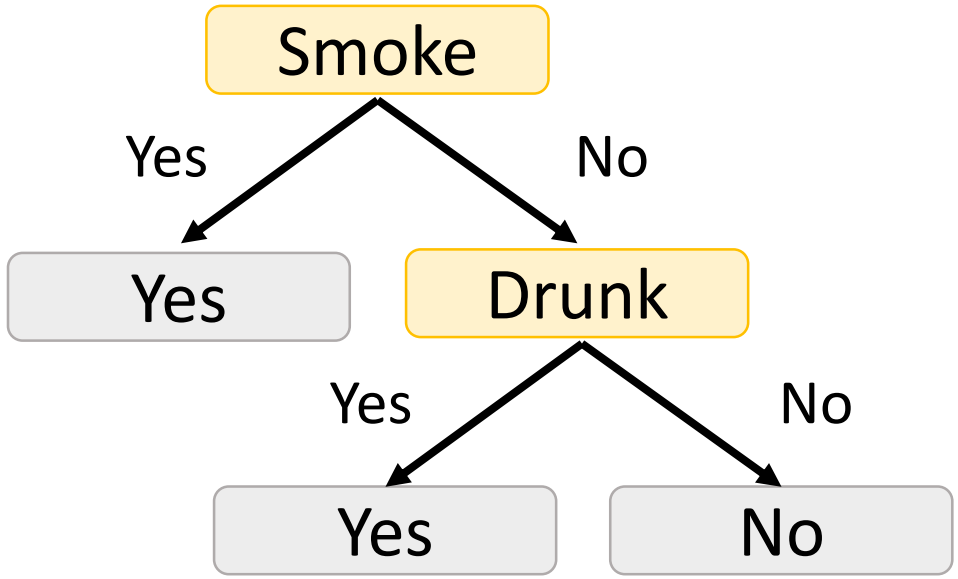
1. **Decision Tree Structure**: Like all decision trees, Regression Trees have a hierarchical, tree-like structure. They consist of nodes, branches, and leaves. Each node represents a decision based on a specific feature, and each leaf represents a predicted value.
2. **Splitting Criteria**: At each node of the tree, the algorithm selects a feature (predictor variable) and a split point to partition the data into two or more subsets. The selection of the feature and split point is based on a criterion that aims to minimize the error in the predicted values. Common splitting criteria include mean squared error or variance reduction.
3. **Predictive Model**: The leaves of the tree contain predicted values for the target variable. In the case of Regression Trees, these are continuous numerical values. The predicted value for a new data point is determined by traversing the tree from the root to an appropriate leaf.
4. **Recursive Partitioning**: Regression Trees are built using a recursive partitioning process. The dataset is repeatedly split into subsets until certain stopping criteria are met. These criteria can include a maximum tree depth, a minimum number of data points in a leaf, or a threshold for error reduction.

4. **Recursive Partitioning**: Regression Trees are built using a recursive partitioning process. The dataset is repeatedly split into subsets until certain stopping criteria are met. These criteria can include a maximum tree depth, a minimum number of data points in a leaf, or a threshold for error reduction.
5. **Pruning**: After constructing the tree, a pruning step can be applied to reduce the complexity and avoid overfitting. Pruning removes branches and nodes that do not significantly improve predictive accuracy on unseen data.
6. **Interpretability**: One of the advantages of Regression Trees is their interpretability. The tree structure is easy to visualize and understand, making it useful for explaining the relationship between features and the target variable.
7. **Handling Nonlinear Relationships**: Regression Trees can capture nonlinear relationships between features and the target variable, which is a limitation of traditional linear regression.
8. **Robust to Outliers**: Regression Trees are relatively robust to outliers in the data, as they are less influenced by extreme values compared to some other regression methods.
9. **Limitations**: Regression Trees can be sensitive to small variations in the data and may overfit when not pruned. They also have limited expressiveness for complex relationships compared to more advanced techniques like Random Forests or Gradient Boosting.

In summary, Regression Trees are a powerful tool for modeling relationships in data with a continuous target variable. They are particularly useful when you want to understand and interpret the decision process and have the flexibility to capture both linear and nonlinear relationships.

Information Gain Example (1)

Holder	Smoke	Drunk	Disease
1	Yes	Yes	Yes
2	Yes	No	Yes
3	Yes	No	Yes
4	No	Yes	Yes
5	No	No	No
6	No	No	No



Information Gain (IG)

- $Entropy$:

$$H = -\sum_{i=1}^{K} p_K \log_2 p_K$$

- $Information\ Gain:$

$$\Delta H = H - \frac{m_L}{m} H_L - \frac{m_R}{m} H_R$$

where m is the total number of instances, with m_K instances belonging to class k, where K = 1, … , k.

Information Gain Example (2)

Holder	Smoke	Drunk	Disease
1	Yes	Yes	Yes
2	Yes	No	Yes
3	Yes	No	Yes
4	No	Yes	Yes
5	No	No	No
6	No	No	No

$$Entropy(Disease) = H = -\sum_{i=1}^{K} p_K \log_2 p_K$$

$$= -\frac{2}{6}\log_2\frac{2}{6} - \frac{4}{6}\log_2\frac{4}{6}$$
$$= 0.918$$

Information Gain Example (3)

Holder	Smoke	Drunk	Disease
1	Yes	Yes	Yes
2	Yes	No	Yes
3	Yes	No	Yes
4	No	Yes	Yes
5	No	No	No
6	No	No	No

$$Entropy(Disease) = H = -\sum_{i=1}^{K} p_K \log_2 p_K$$

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

$$= 0.918$$

$$Information\ Gain(Drunk) = \Delta H = H - \frac{m_L}{m}H_L - \frac{m_R}{m}H_R$$

$$= 0.918 - \frac{2}{6}H_L - \frac{4}{6}H_R$$

Information Gain Example (4)

Holder	Smoke	Drunk	Disease
1	Yes	**Yes**	Yes
2	Yes	No	Yes
3	Yes	No	Yes
4	No	**Yes**	Yes
5	No	No	No
6	No	No	No

$$Entropy(Disease) = H = -\sum_{i=1}^{K} p_K \log_2 p_K$$

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

$$= 0.918$$

$$Information\ Gain(Drunk) = \Delta H = H - \frac{m_L}{m}H_L - \frac{m_R}{m}H_R$$

$$= 0.918 - \frac{2}{6}H_L - \frac{4}{6}H_R$$

$$H_L = -\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2} = 0$$

Information Gain Example (5)

Holder	Smoke	Drunk	Disease
1	Yes	Yes	Yes
2	Yes	**No**	Yes
3	Yes	**No**	Yes
4	No	Yes	Yes
5	No	**No**	No
6	No	**No**	No

$$Entropy(Disease) = H = -\sum_{i=1}^{K} p_K \log_2 p_K$$

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

$$= 0.918$$

$$Information\ Gain(Drunk) = \Delta H = H - \frac{m_L}{m}H_L - \frac{m_R}{m}H_R$$

$$= 0.918 - \frac{2}{6}H_L - \frac{4}{6}H_R$$

$$H_L = -\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2} = 0$$

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

Information Gain Example (6)

Holder	Smoke	Drunk	Disease
1	Yes	Yes	Yes
2	Yes	No	Yes
3	Yes	No	Yes
4	No	Yes	Yes
5	No	No	No
6	No	No	No

$$Entropy(Disease) = H = -\sum_{i=1}^{K} p_K \log_2 p_K$$

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

$$= 0.918$$

$$Information\ Gain(Drunk) = \Delta H = H - \frac{m_L}{m}H_L - \frac{m_R}{m}H_R$$

$$= 0.918 - \frac{2}{6}H_L - \frac{4}{6}H_R = 0.918 - \frac{2}{6} * 0 - \frac{4}{6} * 1 = 0.251$$

$$H_L = -\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2} = 0$$

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

Information Gain Example (7)

Holder	Smoke	Drunk	Disease
1	Yes	Yes	Yes
2	Yes	No	Yes
3	Yes	No	Yes
4	No	Yes	Yes
5	No	No	No
6	No	No	No

$$Entropy(Disease) = H = -\sum_{i=1}^{K} p_K \log_2 p_K$$

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

$$= 0.918$$

$$Information\ Gain(Smoke) = \Delta H = H - \frac{m_L}{m}H_L - \frac{m_R}{m}H_R$$

$$= 0.918 - \frac{3}{6}H_L - \frac{3}{6}H_R$$

Information Gain Example (8)

Holder	Smoke	Drunk	Disease
1	**Yes**	Yes	Yes
2	**Yes**	No	Yes
3	**Yes**	No	Yes
4	No	Yes	Yes
5	No	No	No
6	No	No	No

$$Entropy(Disease) = H = -\sum_{i=1}^{K} p_K \log_2 p_K$$

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

$$= 0.918$$

$$Information\ Gain(Smoke) = \Delta H = H - \frac{m_L}{m}H_L - \frac{m_R}{m}H_R$$

$$= 0.918 - \frac{3}{6}H_L - \frac{3}{6}H_R$$

$$H_L = -\frac{0}{3}\log_2\frac{0}{3} - \frac{3}{3}\log_2\frac{3}{3} = 0$$

Information Gain Example (9)

Holder	Smoke	Drunk	Disease
1	Yes	Yes	Yes
2	Yes	No	Yes
3	Yes	No	Yes
4	**No**	Yes	Yes
5	**No**	No	No
6	**No**	No	No

$$Entropy(Disease) = H = -\sum_{i=1}^{K} p_K \log_2 p_K$$

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

$$= 0.918$$

$$Information\ Gain(Smoke) = \Delta H = H - \frac{m_L}{m}H_L - \frac{m_R}{m}H_R$$

$$= 0.918 - \frac{3}{6}H_L - \frac{3}{6}H_R$$

$$H_L = -\frac{0}{3}\log_2\frac{0}{3} - \frac{3}{3}\log_2\frac{3}{3} = 0$$

$$H_R = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.918$$

Information Gain Example (10)

Holder	Smoke	Drunk	Disease
1	Yes	Yes	Yes
2	Yes	No	Yes
3	Yes	No	Yes
4	No	Yes	Yes
5	No	No	No
6	No	No	No

$$Entropy(Disease) = H = -\sum_{i=1}^{K} p_K \log_2 p_K$$

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

$$= 0.918$$

$$Information\ Gain(Smoke) = \Delta H = H - \frac{m_L}{m}H_L - \frac{m_R}{m}H_R$$

$$= 0.918 - \frac{3}{6}H_L - \frac{3}{6}H_R = 0.918 - \frac{3}{6} * 0 - \frac{3}{6} * 0.918 = 0.459$$

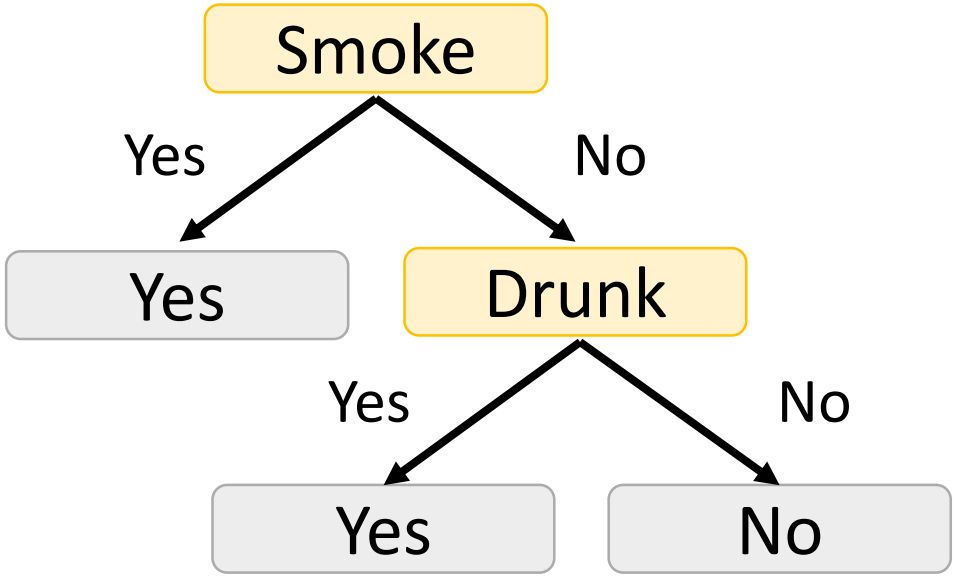
$$H_L = -\frac{0}{3}\log_2\frac{0}{3} - \frac{3}{3}\log_2\frac{3}{3} = 0$$

$$H_R = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.918$$

Information Gain Example (11)

Holder	Smoke	Drunk	Disease
1	Yes	Yes	Yes
2	Yes	No	Yes
3	Yes	No	Yes
4	No	Yes	Yes
5	No	No	No
6	No	No	No

$$Information\ Gain(Drunk) = 0.251$$
$$Information\ Gain(Smoke) = 0.459$$



Tree size

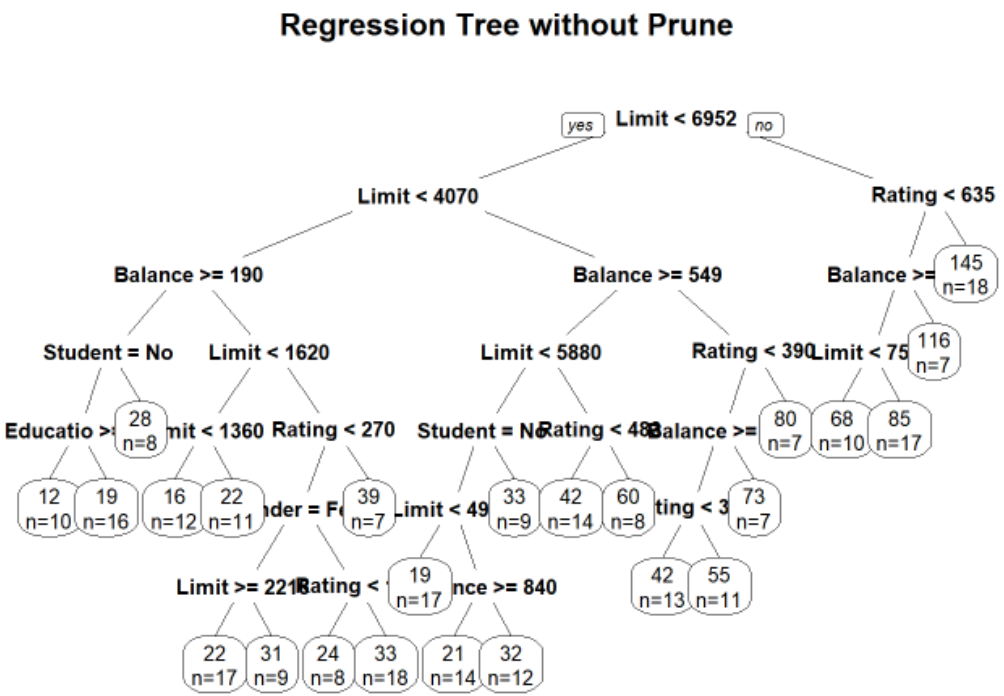
- How large should we grow the tree?
  Fewer splits or fewer regions lower variance better interpretation at cost of little more bias
- Tree size is a tuning parameter governing the model's complexity, and the optimal tree size should be adaptively chosen from the data.
- Idea? Stop splitting when RSS improvement is lower than a threshold
  - Might not be good since a split early on in the tree might be followed by a very good split; a split that leads to a large reduction in RSS later on

Pruning

- Pruning is a technique in machine learning that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances. Pruning reduces the complexity of the final classifier and hence improves predictive accuracy by the reduction of overfitting.

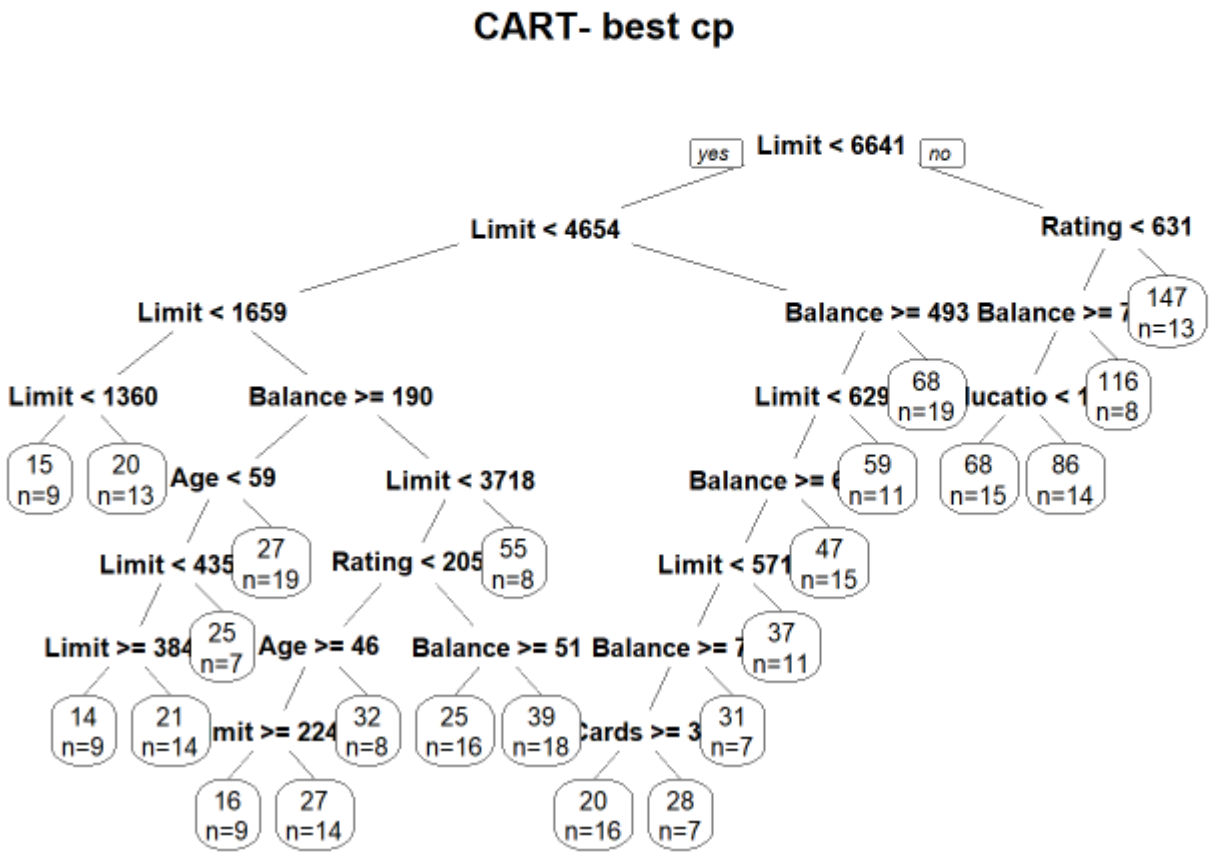
Demo of Regression Trees without Prue

```
# Grow regression trees
# arguments:
# cp: complexity parameter is the tuning parameter in CART.
#Any split that does not decrease the overall lack of fit by
# a factor of cp is not attempted.
# Used to choose depth of the tree, we'll manually prune the tree later and
# hence can set the threshold very low.
# method: "anova" for regression tree, "class" for classification tree

#Fitting the model
fittree <- rpart(Income~., data=train, method="anova", cp=10^(-5))
prp(fittree, faclen=-3, extra=1, cex=.8, main="Regression Tree without Prune")
```



Demo of Regression Trees with Prue

```
#identify best cp value to use
bestcp <- fittree$cptable[which.min(fittree$cptable[,"xerror"]),"CP"]
fittreebest <- prune(fittree, cp=bestcp)
# Plot tree using package rpart.plot
prp(fittreebest, faclen=-3, extra=1, cex=.8, main="CART- best cp")
```

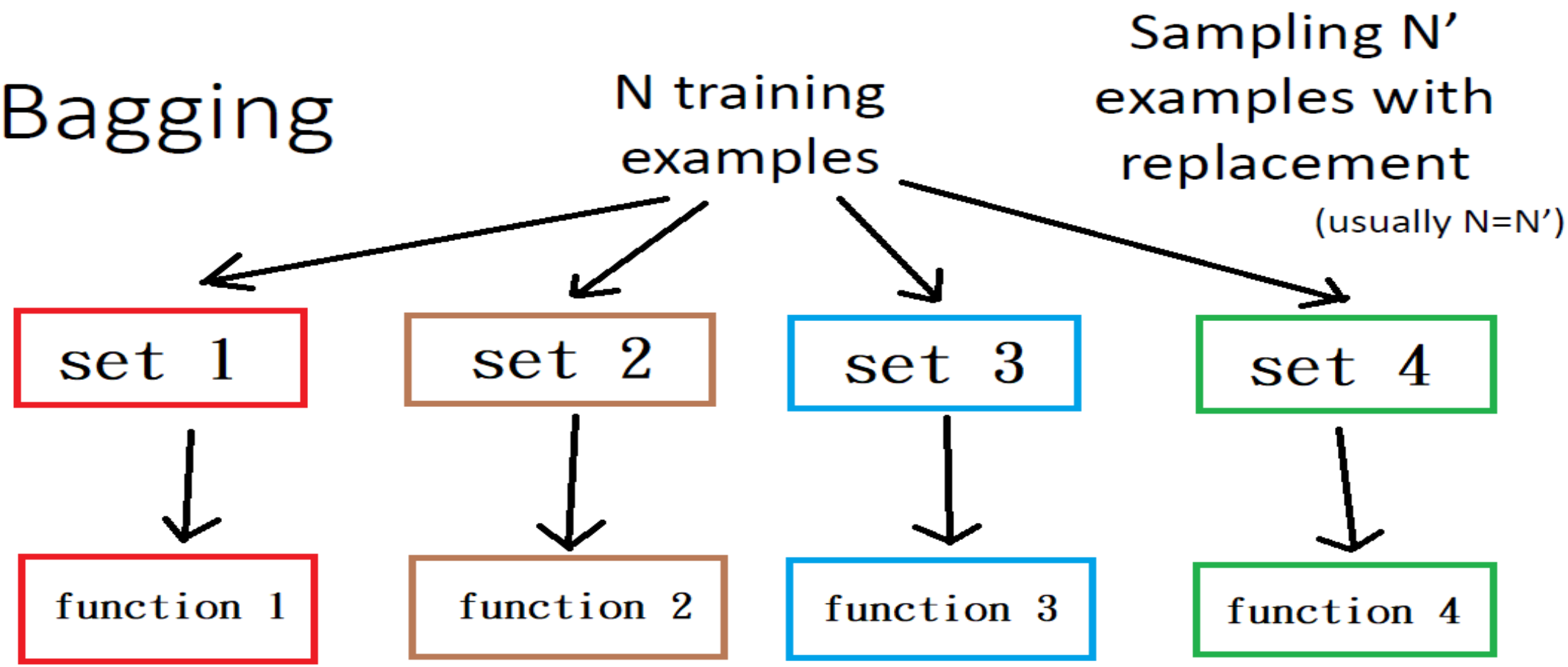


Ensemble learning

Bagging

- **Bagging** is a technique for reducing the variance of an estimated prediction function.
  - Simply fit the same prediction function many times to **bootstrap** sampled versions of the training data, and average the result
- The essential idea is to average many noisy but approximately unbiased models, and hence reduce the variance.

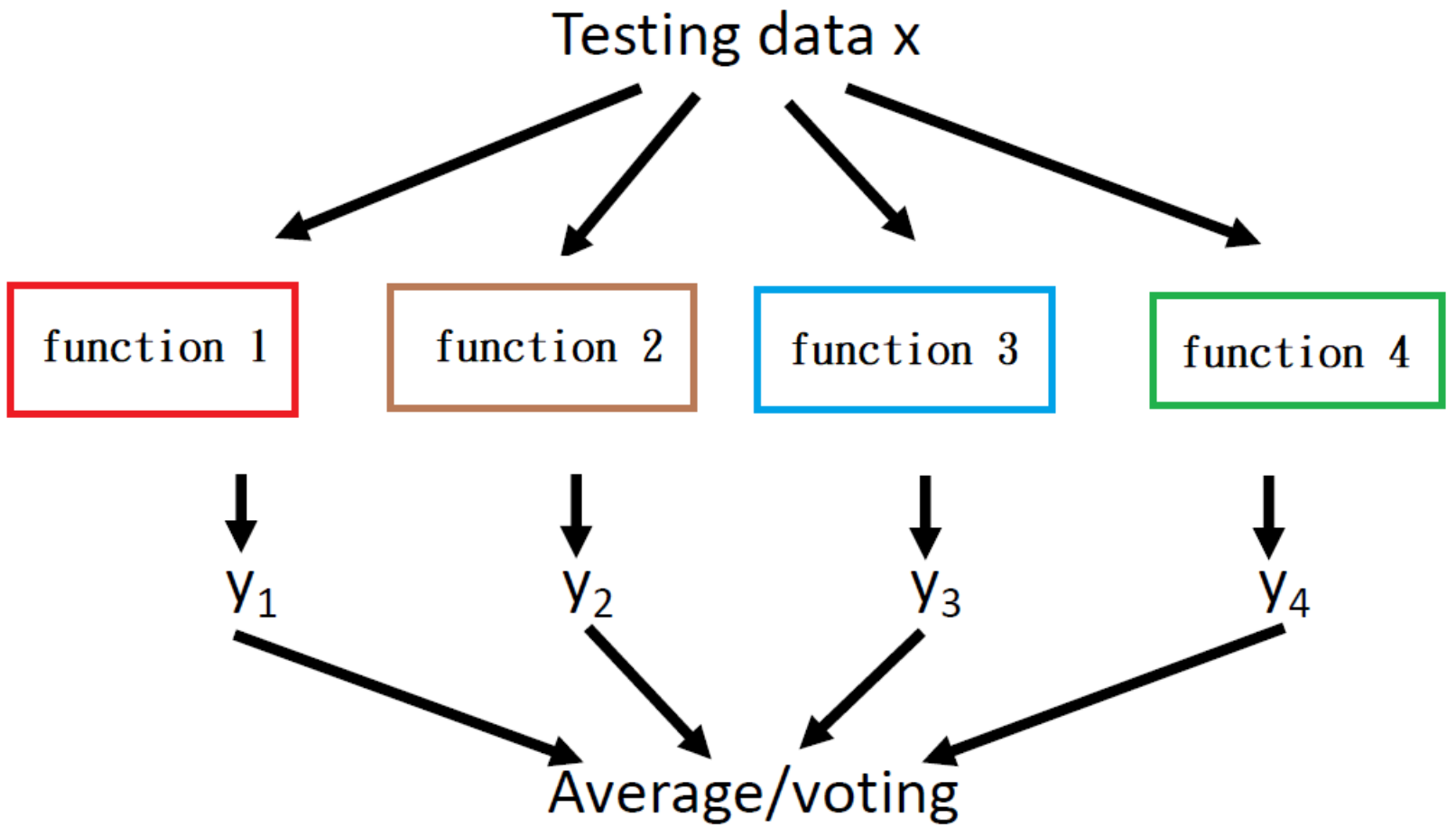
Bagging


N training examples
Sampling N' examples with replacement
(usually N=N')
set 1
set 2
set 3
set 4
function 1
function 2
function 3
function 4

Bagging

This approach would be helpful when your model is complex, easy to overfit.

e.g. decision tree



Random forests

- Trees are ideal candidates for bagging:
  - Trees can capture complex interaction structures in the data, and if grown sufficiently deep, have relatively low bias.
  - Trees are notoriously noisy, they benefit greatly from the averaging.
- **Random forests** is a substantial modification of bagging that builds a large collection of *de-correlated* trees, and then averages them.

Random forests: algorithm

Training data: (x_i, y_i), $i = 1, 2, \cdots, N$, with $x_i = (x_{i1}, x_{i2}, \cdots, x_{ip})$ and y_i is the response

1. For $b = 1$ to B
   - 1.1 Draw a bootstrap sample: sample N cases at random with replacement from the original training data
   - 1.2 Grow a tree T_b to the bootstrapped sample, by recursively repeating the following steps for each node split, until the minimum node size n_{min} is reached.
     - 1.2.1 Select m variables at random from the p variables
     - 1.2.2 Pick the best variable/split-point among the m
     - 1.2.3 Split the node into two daughter nodes
   - 1.3 There is no pruning.
2. Output the ensemble of trees $\{T_b\}_1^B$

- To make a prediction at a new point x :
  - *Regression*: $\hat{f}_{rf}^{B} = \frac{1}{B}\sum_{b=1}^{B} T_b(\boldsymbol{x})$
  - *Classification*: Let $\hat{C}_b(\boldsymbol{x})$ be the class prediction of the bth bootstrap tree. Then $\hat{C}_{rf}^{B} =$ majority vote$\{\hat{C}_b(\boldsymbol{x})\}_1^B$

Out-of-bag samples

- An important feature of random forests is its use of out-of-bag (OOB) samples:
  - For each observation $\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$, construct its random forest predictor by averaging only those trees corresponding to bootstrap samples in which $\boldsymbol{z}_i$ did not appear.
- An OOB error estimate is almost identical to that obtained by N-fold cross-validation
- Unlike many other nonlinear estimators, random forests can be fit in one sequence, with cross-validation being performed along the way. Once the OOB error stabilizes, the training can be terminated.

Random Forest

train	f_1	f_2	f_3	f_4
x^1	O	X	O	X
x^2	O	X	X	O
x^3	X	O	O	X
x^4	X	O	X	O

- Decision tree:
  - Easy to achieve 0% error rate on training data
    - If each training example has its own leaf .....
- Random forest: Bagging of decision tree
  - Resampling training data is not sufficient
  - Randomly restrict the features/questions used in each split
- Out-of-bag validation for bagging
  - Using RF = f_2+f_4 to test x^1
  - Using RF = f_2+f_3 to test x^2
  - Using RF = f_1+f_4 to test x^3
  - Using RF = f_1+f_3 to test x^4

Out-of-bag (OOB) error

Good error estimation of testing set

Why random selection of the input variables?

- An average of B i.i.d. random variables, each with variance σ^2, has variance $\frac{1}{B}\sigma^2$.
- If the variables are simply i.d. (identically distributed, but not necessarily independent) with positive pairwise correlation ρ, the variance of the average is $\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$.
  - As B increases, the second term disappears, but the first remains.
- Hence **the size of the correlation of pairs of bagged trees limits the benefits of averaging**.
- The idea in random forests is to improve the variance reduction of bagging by reducing the correlation between the trees– **This is achieved in the tree-growing process through random selection of the input variables**.

Boosting

Training data:

$\{(x^1, \hat{y}^1), \cdots, (x^n, \hat{y}^n), \cdots, (x^N, \hat{y}^N)\}$

$\hat{y} = \pm 1$ (binary classification)

- Guarantee:
  - If your ML algorithm can produce classifier with error rate smaller than 50% on training data
  - You can obtain 0% error rate classifier after boosting.
- Framework of boosting
  - Obtain the first classifier $f_1(x)$
  - Find another function $f_2(x)$ to help $f_1(x)$
    - However, if $f_2(x)$ is similar to $f_1(x)$, it will not help a lot.
    - We want $f_2(x)$ to be complementary with $f_1(x)$ (How?)
  - Obtain the second classifier $f_2(x)$
  - ..... Finally, combining all the classifiers
- The classifiers are learned sequentially.

How to obtain different classifiers?

- Training on different training data sets
- How to have different training data sets
  - Re-sampling your training data to form a new set
  - Re-weighting your training data to form a new set
  - In real implementation, you only have to change the cost/objective function

$(x^1, \hat{y}^1, u^1)$ $u^1 = 1 \rightarrow 0.4$

$(x^2, \hat{y}^2, u^2)$ $u^2 = 1 \rightarrow 2.1$

$(x^3, \hat{y}^3, u^3)$ $u^3 = 1 \rightarrow 0.7$

$$L(f) = \sum_n l(f(x^n), \hat{y}^n)$$

$$L(f) = \sum_n u^n l(f(x^n), \hat{y}^n)$$

Idea of Adaboost

- Idea: **training $f_2(x)$ on the new training set that fails $f_1(x)$**
- How to find a new training set that fails $f_1(x)$?

ε_1: the error rate of $f_1(x)$ on its training data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n \qquad \varepsilon_1 < 0.5$$

Changing the example weights from u_1^n to u_2^n such that

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

The performance of f_1 for new weights would be random.

Training $f_2(x)$ based on the new weights u_2^n

Re-weighting Training Data

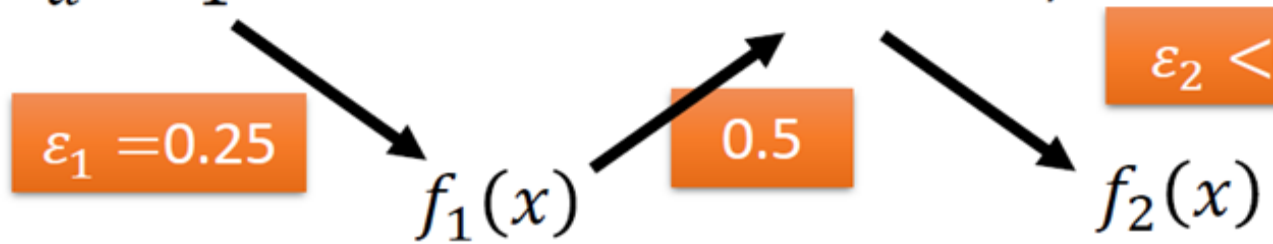
- Idea: **training $f_2(x)$ on the new training set that fails $f_1(x)$**
- How to find a new training set that fails $f_1(x)$?

$(x^1, \hat{y}^1, u^1)$ $u^1 = 1$ Predict Correct $u^1 = 1/\sqrt{3}$ Decrease Weights

$(x^2, \hat{y}^2, u^2)$ $u^2 = 1$ Predict Wrong $u^2 = \sqrt{3}$ Increase Weights

$(x^3, \hat{y}^3, u^3)$ $u^3 = 1$ Predict Correct $u^3 = 1/\sqrt{3}$ Decrease Weights

$(x^4, \hat{y}^4, u^4)$ $u^4 = 1$ Predict Correct $u^4 = 1/\sqrt{3}$ Decrease Weights



Re-weighting Training Data

- Idea: **training $f_2(x)$ on the new training set that fails $f_1(x)$**
- How to find a new training set that fails $f_1(x)$?

If x^n misclassified by f_1 ($f_1(x^n) \neq \hat{y}^n$)

$u_2^n \leftarrow u_1^n$ multiplying d_1 increase

If x^n correctly classified by f_1 ($f_1(x^n) = \hat{y}^n$)

$u_2^n \leftarrow u_1^n$ devided by d_1 decrease

f_2 will be learned based on example weights u_2^n

What is the value of d_1?

$$\epsilon_1 = \frac{\Sigma_n u_1^{(n)} \delta(f_1(x^{(n)}) \neq \hat{y}^{(n)})}{Z_1} < 0.5, Z_1 = \Sigma_n u_1^{(n)}$$

$$\rightarrow \epsilon_2 = \frac{\Sigma_n u_2^{(n)} \delta(f_1(x^{(n)}) \neq \hat{y}^{(n)})}{Z_2} = 0.5, Z_2 = \Sigma_n u_2^{(n)}$$

if $f_1(x^{(n)}) \neq \hat{y}^{(n)}, u_1^{(n)} * d_1 = u_2^{(n)}$

if $f_1(x^{(n)}) = \hat{y}^{(n)}, u_1^{(n)} / d_1 = u_2^{(n)}$

$$\Sigma_n u_2^{(n)} \delta(f_1(x^{(n)}) \neq \hat{y}^{(n)}) = \Sigma_{f_1(x^{(n)}) \neq \hat{y^{(n)}}} u_1^{(n)} * d_1$$

$$Z_2 = \Sigma_{f_1(x^{(n)}) \neq \hat{y^{(n)}}} u_1^{(n)} * d_1 + \Sigma_{f_1(x^{(n)}) = \hat{y^{(n)}}} u_1^{(n)} / d_1$$

$$\epsilon_2 = \frac{\Sigma_{f_1(x^{(n)}) \neq \hat{y^{(n)}}} u_1^{(n)} * d_1}{\Sigma_{f_1(x^{(n)}) \neq \hat{y^{(n)}}} u_1^{(n)} * d_1 + \Sigma_{f_1(x^{(n)}) = \hat{y^{(n)}}} u_1^{(n)} / d_1} = 0.5 \Rightarrow$$

$$\Sigma_{f_1(x^{(n)}) \neq \hat{y^{(n)}}} u_1^{(n)} * d_1 = \Sigma_{f_1(x^{(n)}) = \hat{y^{(n)}}} u_1^{(n)} / d_1 \Rightarrow$$

$$d1 \Sigma_{f_1(x^{(n)}) \neq \hat{y^{(n)}}} u_1^{(n)} = \frac{1}{d_1} \Sigma_{f_1(x^{(n)}) = \hat{y^{(n)}}} u_1^{(n)}$$

because $\epsilon_1 = \frac{\Sigma_{f_1(x^{(n)}) \neq \hat{y^{(n)}}} u_1^{(n)}}{Z_1} \Rightarrow \Sigma_{f_1(x^{(n)}) \neq \hat{y^{(n)}}} u_1^{(n)} = \epsilon_1 Z_1$

so $d_1 \epsilon_1 Z_1 = \frac{1}{d_1}(1 - \epsilon_1) Z_1 \Rightarrow d_1 = \sqrt{\frac{1-\epsilon_1}{\epsilon_1}} > 1$

because $\epsilon_1 < 0.5$

Algorithm for AdaBoost

- Giving training data
  $\{(x^1, \hat{y}^1, u_1^1), \cdots, (x^n, \hat{y}^n, u_1^n), \cdots, (x^N, \hat{y}^N, u_1^N)\}$
  - $\hat{y} = \pm 1$ (Binary classification), $u_1^n = 1$ (equal weights)
- For t = 1, …, T:
  - Training weak classifier $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - ε_t is the error rate of $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - For n = 1, …, N:
    - If x^n is misclassified by $f_t(x)$: $\hat{y}^n \neq f_t(x^n)$
    - $u_{t+1}^n = u_t^n \times d_t = u_t^n \times \exp(\alpha_t)$
    - Else:
    - $u_{t+1}^n = u_t^n / d_t = u_t^n \times \exp(-\alpha_t)$

$$d_t = \sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$u_{t+1}^n \leftarrow u_t^n \times exp(-\hat{y}^n f_t(x^n)\alpha_t)$$

Algorithm for AdaBoost

- We obtain a set of functions: $f_1(x), \ldots, f_t(x), \ldots, f_T(x)$
- How to aggregate them?
  - Uniform weight:
    - $H(x) = sign(\sum_{t=1}^{T} f_t(x))$
  - Non-uniform weight:
    - $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

Smaller error ε_t, larger weight for final voting

$$\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$u_{t+1}^n = u_t^n \times exp(-\hat{y}^n f_t(x^n)\alpha_t)$$

$\epsilon^t = 0.1$ $\alpha^t = 1.10$

$\epsilon^t = 0.4$ $\alpha^t = 0.20$

Homework 6 (submitted to e3.nycu.edu.tw before Nov 1, 2023)

- Use R, Python, and suitable computer packages to perform different types of Regression Models (Ex:-Parametric and non-parametric regression).
- Explain the results you obtain.
- Compare the results you obtain by different regression approaches.
- Discuss possible problems you plan to investigate for future studies

Possible sources of open datasets:

- UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets.php)
- Kaggle Datasets (https://www.kaggle.com/datasets)
- World Health Organization Datasets (https://www.who.int/)