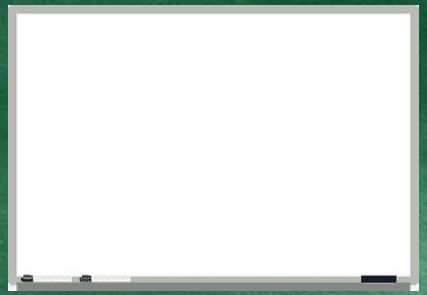
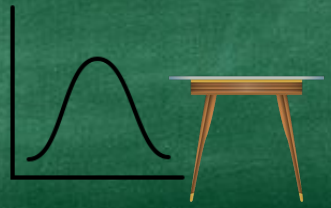
# Data300 - Statistical and Machine Learning



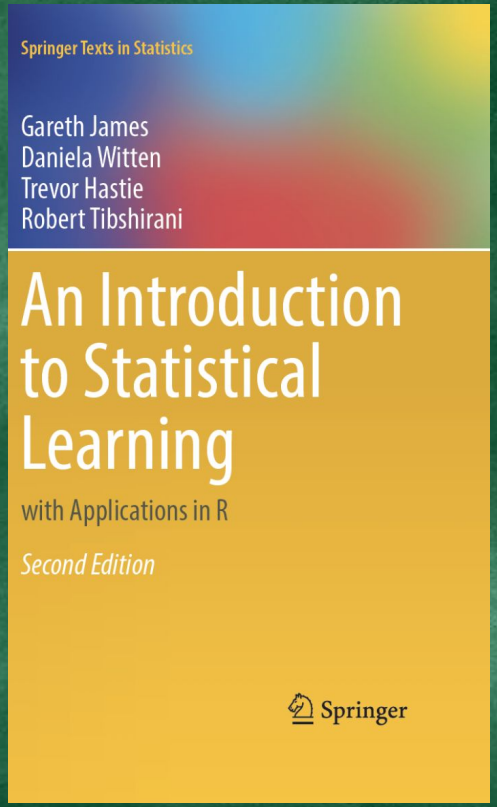
# Recap:

- Linear Regression
- Logistic Regression
- Time Series
- Variable Selection
- Decision Trees
- Support Vector Machine

Special Thanks to:

- Prof. Xiexin Liu / Lulu Wang
  - For the class slides and assignments
    Which we used to put together these slides

- This book
  - For being free online
  - For diagrams and Clear explanations



# Learning objectives:

- Equip machines to learn models from data
- Apply algorithms to real-world problems and report outcomes
- Understand various learning algorithms and optimization
- Evaluate models from data

# Background: Assumptions

Generally speaking, a supervised learning model assumes that there is the following relationship between the predictors X and the response y:

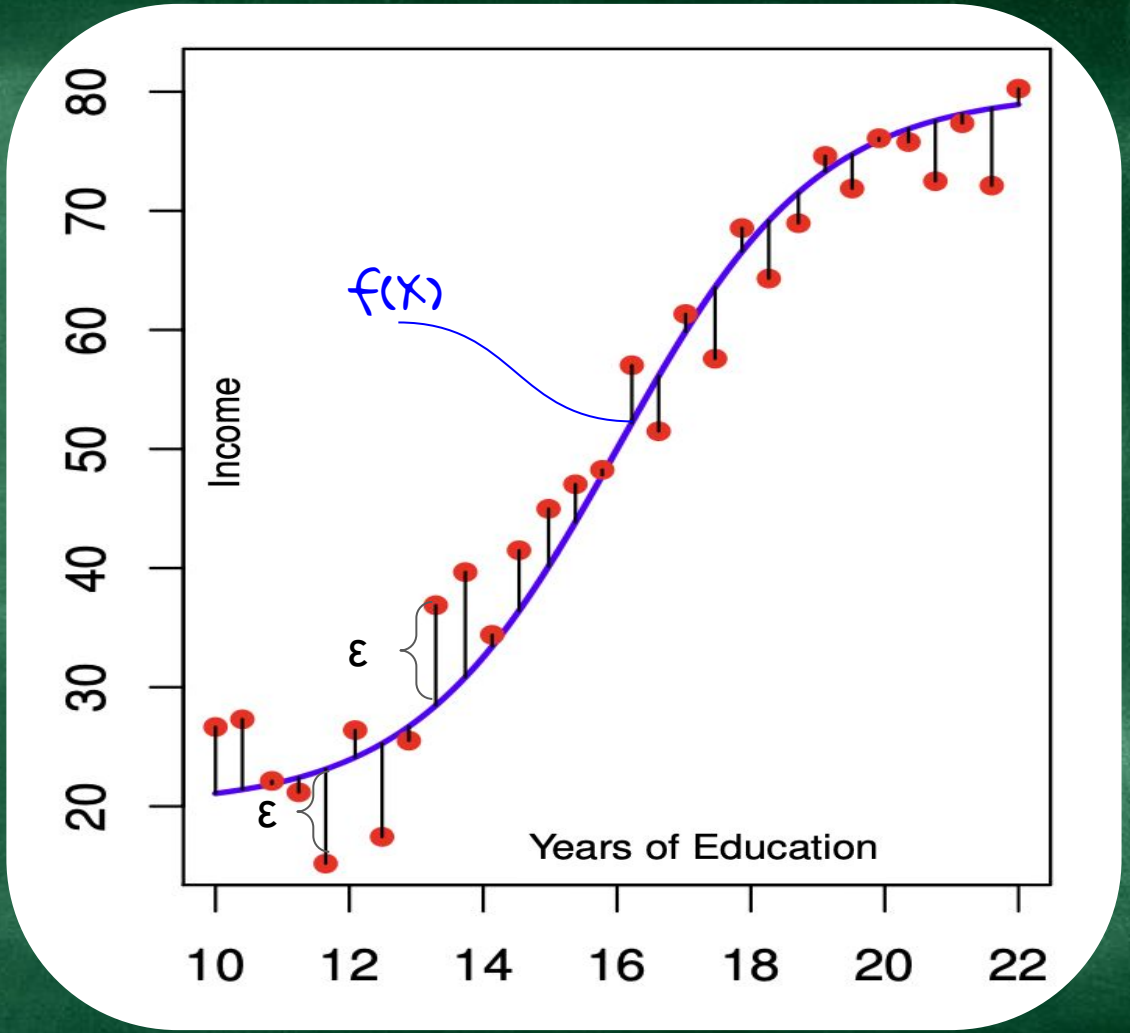
$$y = f(X) + \varepsilon$$

Error term, unique for each y

Response (Y) = Some function of the chosen predictors + Variability of Y not accounted for by F(X)

# Background: Visual

* error terms average to 0



# Background: The goal

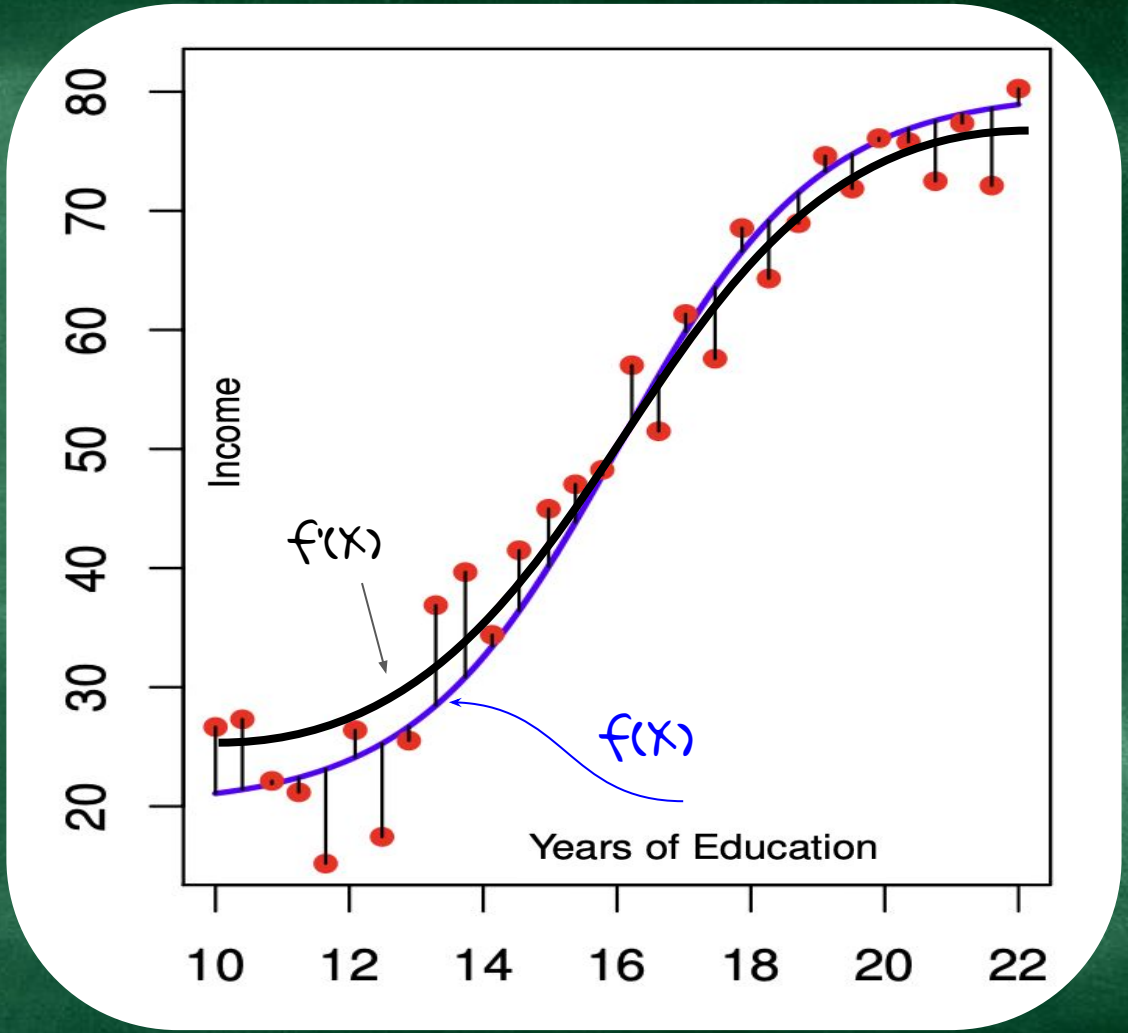
Our goal is to estimate f(X) by assuming that f(X) has some form (parametric, ex: linear equation) such that

$$\hat{y} = \hat{f}(X)$$

Resulting estimate of y

Estimate of f(X)

# Background: Estimated v Actual

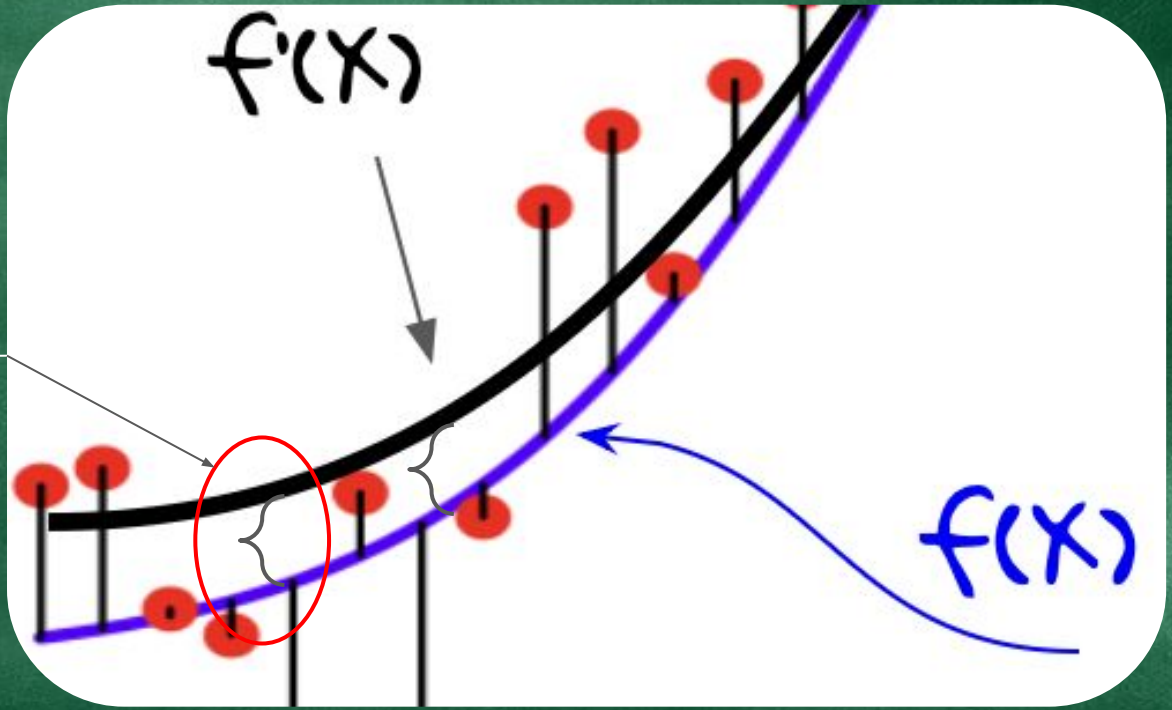


# Background: actual - fitted

We want to reduce this:

f(x) - f'(x),

Or: actual-fitted



# Background: MSE

In regression, the metric used to measure quality of fit is Mean Squared Error

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

Just means: average [ (actual - fitted)$^2$ ]

# Background: MSE Expansion, Bias/Variance

MSE = $E\left[\left(y_0 - \hat{f}(x_0)\right)^2\right]$


$$
\begin{aligned}
&= E\left[(y_0)^2 + \left(\hat{f}(x_0)\right)^2 - 2\left(y_0\hat{f}(x_0)\right)\right] && \text{by expanding} \\
&= E[(y_0)^2] + E\left[\left(\hat{f}(x_0)\right)^2\right] - E\left[2\left(y_0\hat{f}(x_0)\right)\right] && \text{by separating terms} \\
&= Var[y_0] + (E[y_0])^2 + Var[\hat{f}(x_0)] + \left(E[\hat{f}(x_0)]\right)^2 - 2E[y_0\hat{f}(x_0)] && \text{by the variance equation} \\
&= \left(E[y_0] - E[\hat{f}(x_0)]\right)^2 + Var[y_0] + Var[\hat{f}(x_0)] && \text{by combining E[] terms} \\
&= \left[Bias\left(\hat{f}(x_0)\right)\right]^2 + Var[y_0] + Var[\hat{f}(x_0)] && \text{substitute bias} \\
&= \left[Bias\left(\hat{f}(x_0)\right)\right]^2 + Var[\epsilon] + Var[\hat{f}(x_0)] && \text{substitute epsilon}
\end{aligned}
$$

# Background: MSE Expansion, Bias/Variance

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2 + \mathrm{Var}(\epsilon).$$

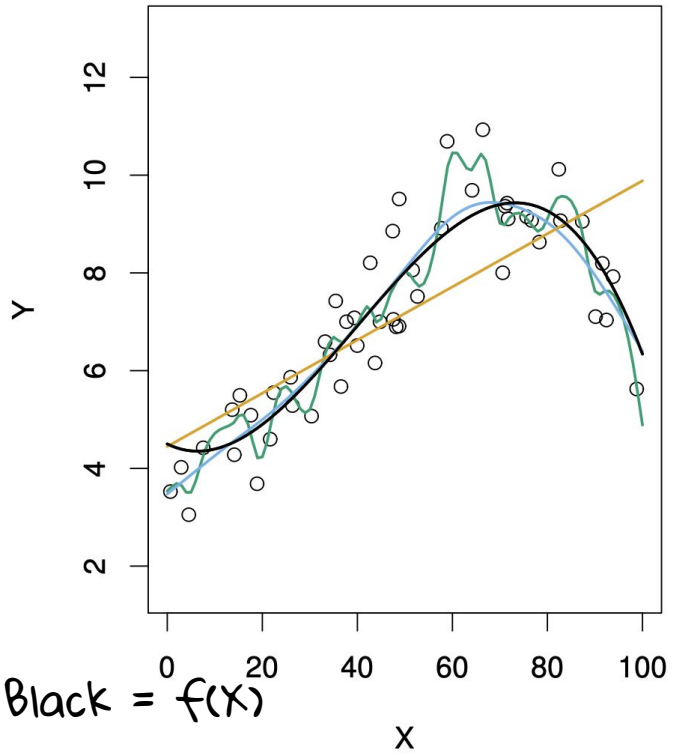
Variance of a model:

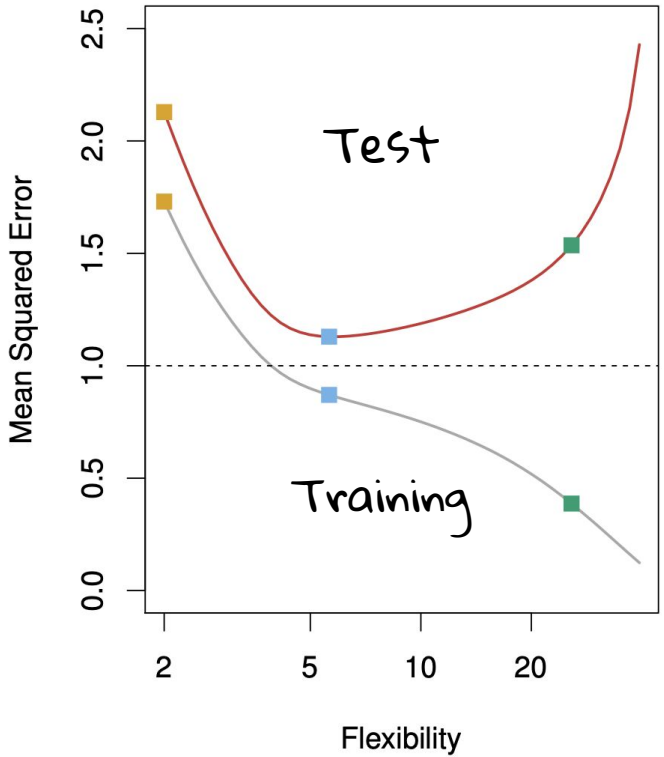
The amount of change in the model when we change the training set

Bias of a model:

The error introduced by using a model to approximate a real-life problem.

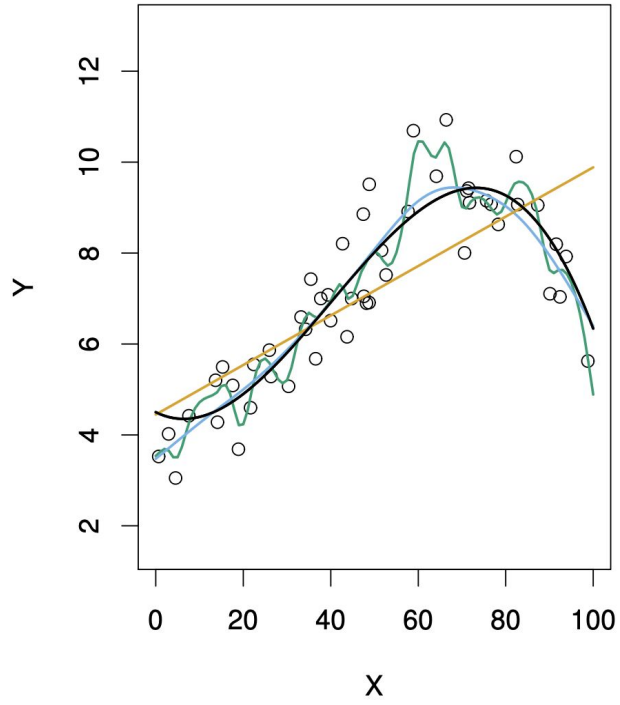
# Background: Trade-off

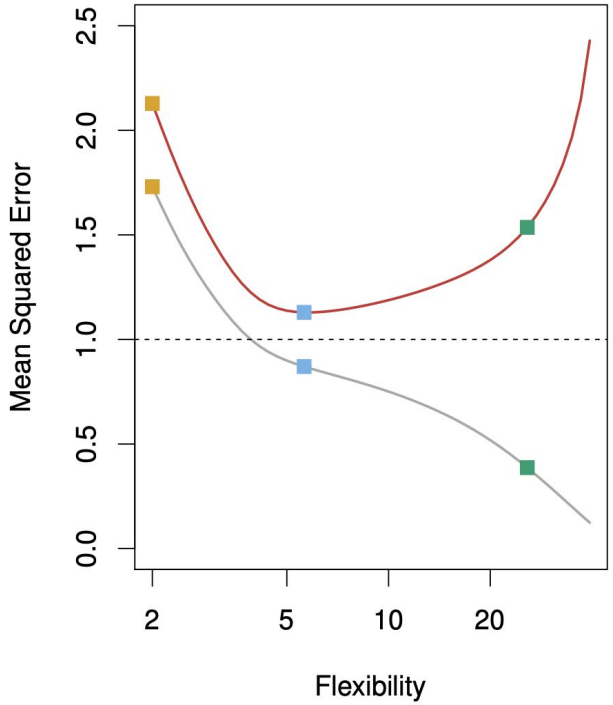


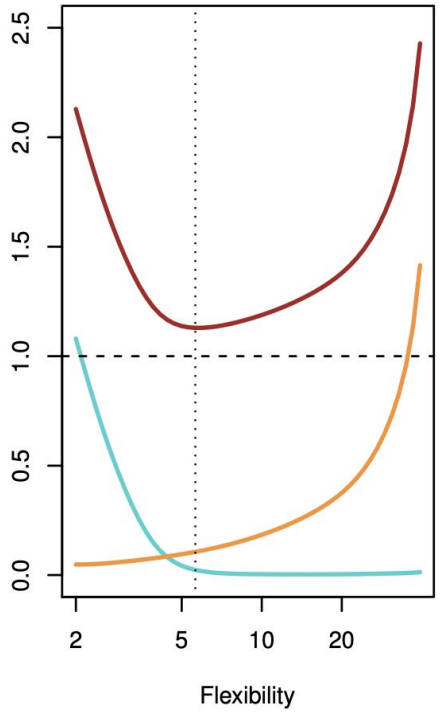




Is the model complex enough to accurately represent the data (Bias) without overfitting the training set (Variace)

# Background: Trade-off



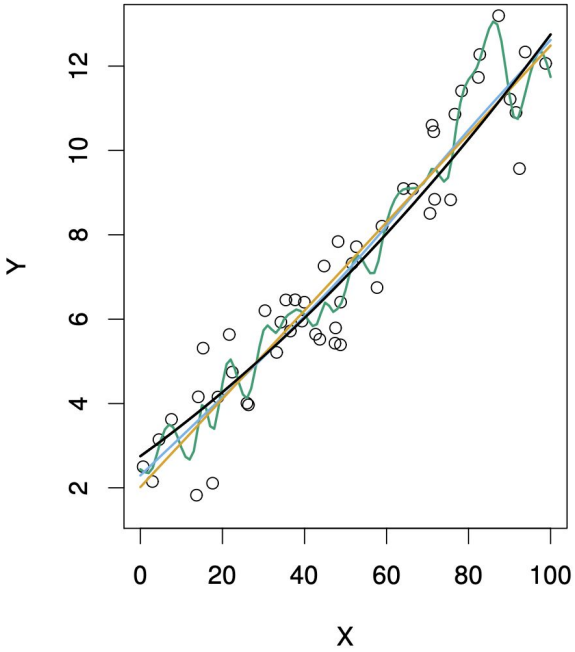



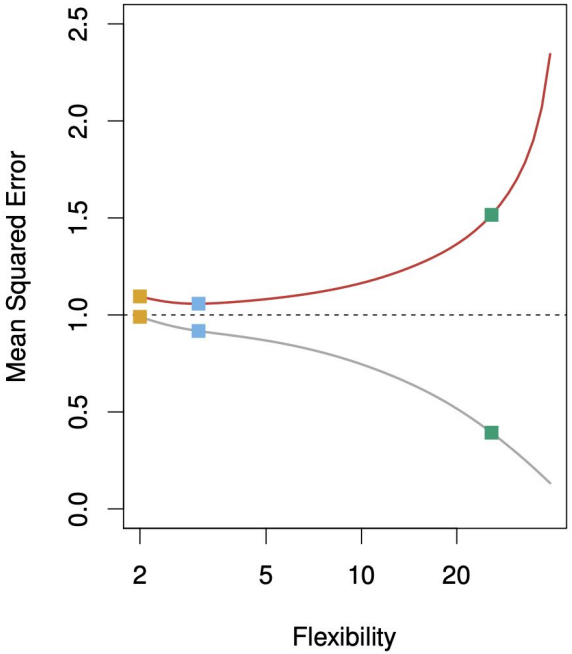



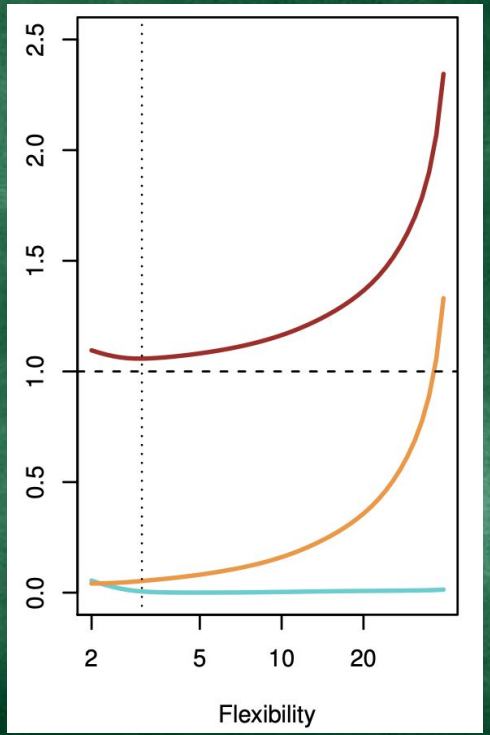


# Background: Trade-off

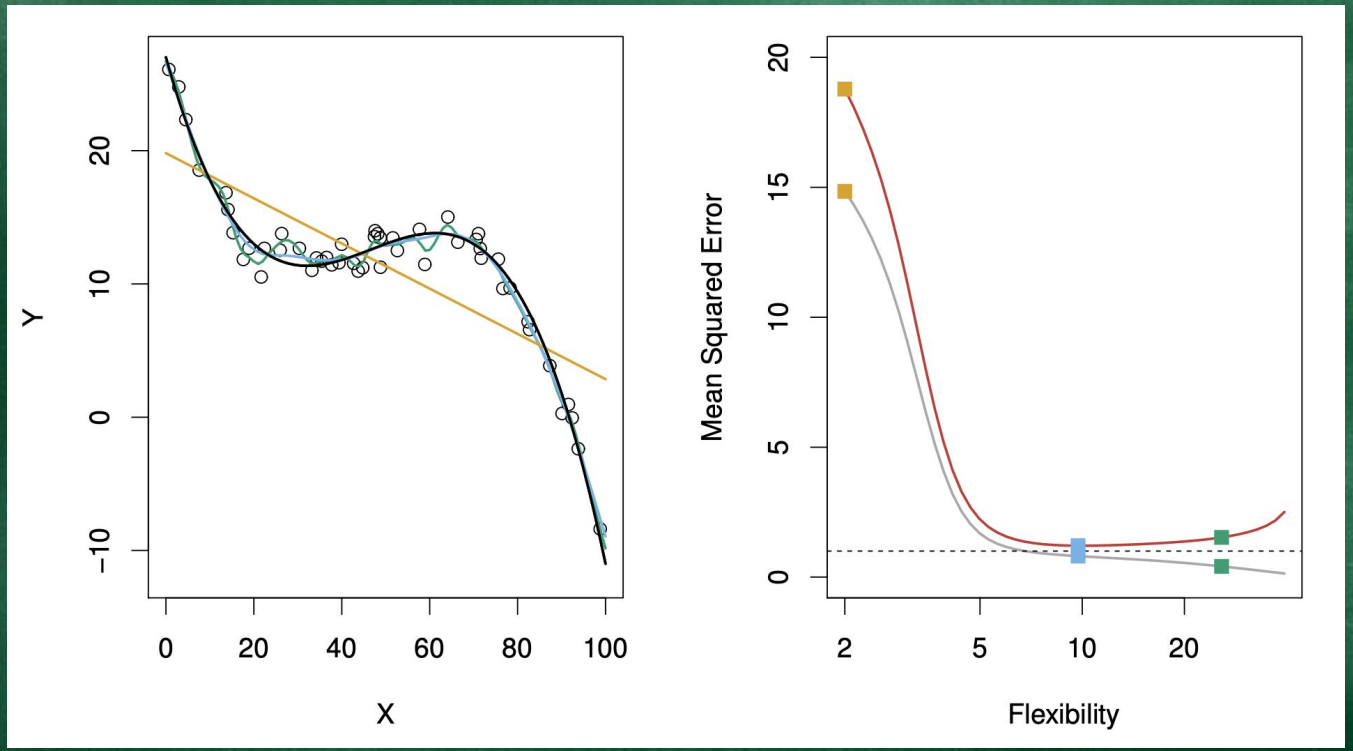
MSE
Bias
Var




# Background: Trade-off



# Linear Regression: Big picture

LR assumes a linear relationship between the predictors and the response, or

$$y = f(X) + \varepsilon$$

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_d x_d$$

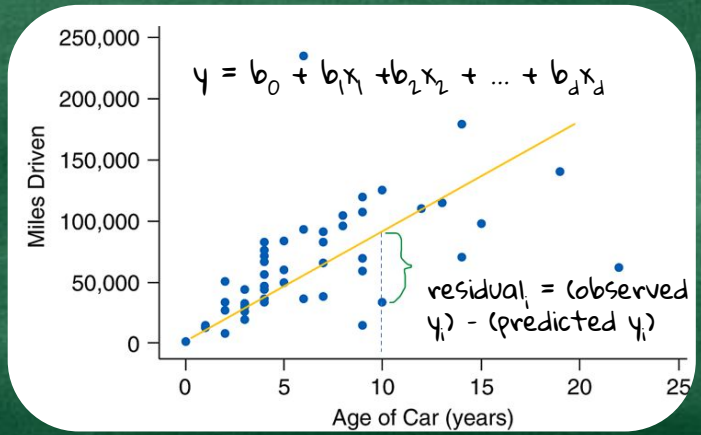
We will estimate each $\beta_n$ to get

$$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_d x_d$$

# Linear Regression: OLS Brief

To estimate the $\beta_n$ we will use Ordinary Least Squared (OLS) method


We minimize (partial derivative = 0) the sum of the the $(\text{residuals})^2$



# Linear Regression: OLS Brief

Example of estimates via OLS with 1 predictor:

residual for unit $i = y_i - \hat{y}_i$

$= y_i - (b_0 + b x_i)$

$$SSR = \sum_{i=1}^{N} (y_i - (b_0 + b x_i))^2$$

$$\frac{\partial SSR}{\partial b_0} = \sum_{i=1}^{N} 2(y_i - (b_0 + b x_i)) \cdot (-1) = 0$$

$$\Rightarrow \sum_{i=1}^{N} [(y_i - b x_i) - b_0] = 0$$

$$\Rightarrow N b_0 = \sum_{i=1}^{N} (y_i - b x_i)$$

$$\Rightarrow b_0 = \frac{1}{N}\left(\sum_{i=1}^{N} y_i - b \sum_{i=1}^{N} x_i\right) = \bar{y} - b\bar{x}$$


$$\frac{\partial SSR}{\partial b} = \sum_{i=1}^{N} 2\left(y_i - (\bar{y} + b(x_i - \bar{x}))\right) \cdot (x_i - \bar{x})(-1) = 0$$

$$\Rightarrow \sum_{i=1}^{N} \left((y_i - \bar{y}) - b(x_i - \bar{x})\right)(x_i - \bar{x}) = 0$$

$$\Rightarrow b \sum_{i=1}^{N} (x_i - \bar{x})^2 = \sum_{i=1}^{N} (y_i - \bar{y})(x_i - \bar{x})$$

$$\Rightarrow b = \frac{\sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{N} (x_i - \bar{x})^2}$$

# LR output

Contents are from DataCamp guided project.

- Candy rankings
- Winpercent: the percentage of people who prefer this candy over another randomly chosen candy from the dataset

| competitorname | chocolate | fruity | caramel | peanutyalmondy | nougat | crispedricewafer | hard | bar | pluribus | sugarpercent | pricepercent | winpercent |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| <chr> | <lgl> | <lgl> | <lgl> | <lgl> | <lgl> | <lgl> | <lgl> | <lgl> | <lgl> | <dbl> | <dbl> | <dbl> |
| 100 Grand | TRUE | FALSE | TRUE | FALSE | FALSE | TRUE | FALSE | TRUE | FALSE | 0.732 | 0.860 | 66.97173 |
| 3 Musketeers | TRUE | FALSE | FALSE | FALSE | TRUE | FALSE | FALSE | TRUE | FALSE | 0.604 | 0.511 | 67.60294 |
| One dime | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | 0.011 | 0.116 | 32.26109 |
| One quarter | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | 0.011 | 0.511 | 46.11650 |
| Air Heads | FALSE | TRUE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | 0.906 | 0.511 | 52.34146 |
| Almond Joy | TRUE | FALSE | FALSE | TRUE | FALSE | FALSE | FALSE | TRUE | FALSE | 0.465 | 0.767 | 50.34755 |

```
win_mod <- lm(winpercent ~ .-competitorname, candy_rankings)
```

# LR output

Estimate:
- estimate for $\beta_n$ (ie. the coefficients or $b_n$)

Std. Error:
- the average amount this estimate ($b_n$) deviates from the actual value ($\beta_n$)

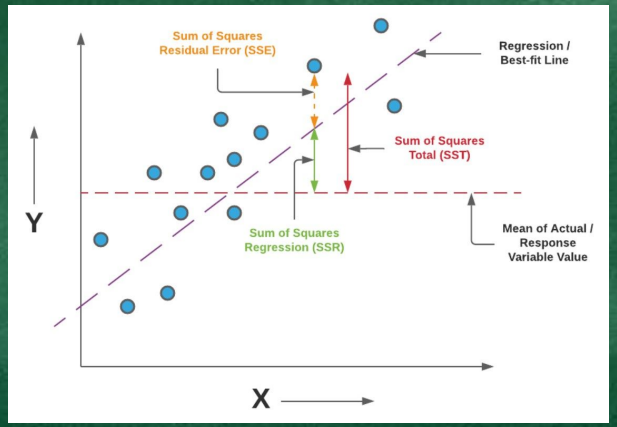
P-value:
- The probability of observing this data if $\beta_n = 0$ (ie. if this variable has no effect on the response)

```
Call:
lm(formula = winpercent ~ . - competitorname, data = candy_rankings)

Residuals:
     Min       1Q   Median       3Q      Max
-20.2244  -6.6247   0.1986   6.8420  23.8680

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         34.5340     4.3199   7.994 1.44e-11 ***
chocolateTRUE       19.7481     3.8987   5.065 2.96e-06 ***
fruityTRUE           9.4223     3.7630   2.504  0.01452 *
caramelTRUE          2.2245     3.6574   0.608  0.54493
peanutyalmondyTRUE  10.0707     3.6158   2.785  0.00681 **
nougatTRUE           0.8043     5.7164   0.141  0.88849
crispedricewaferTRUE 8.9190     5.2679   1.693  0.09470 .
hardTRUE            -6.1653     3.4551  -1.784  0.07852 .
barTRUE              0.4415     5.0611   0.087  0.93072
pluribusTRUE        -0.8545     3.0401  -0.281  0.77945
sugarpercent         9.0868     4.6595   1.950  0.05500 .
pricepercent        -5.9284     5.5132  -1.075  0.28578
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.7 on 73 degrees of freedom
Multiple R-squared:  0.5402,    Adjusted R-squared:  0.4709
F-statistic: 7.797 on 11 and 73 DF,  p-value: 9.504e-09
```

# LR output

R-squared:

- percentage of the variation in the response variable that is explained by the linear regression model




```
Residual standard error: 10.7 on 73 degrees of freedom
Multiple R-squared:  0.5402,    Adjusted R-squared:  0.4709
F-statistic: 7.797 on 11 and 73 DF,  p-value: 9.504e-09
```

F-statistic p-value:

- The probability of observing this data if all the $\beta_n = 0$
- at least 1 coefficient is not 0

# LR Interpretations

| | | | | | |
|---|---|---|---|---|---|
| chocolateTRUE | 19.7481 | 3.8987 | 5.065 | 2.96e-06 | *** |
| sugarpercent | 9.0868 | 4.6595 | 1.950 | 0.05500 | . |

Estimate:

- A 1 percent increase sugar-percent is associated with a 9.0898 increase in winpercent on average holding all other predictors fixed.
- Being a chocolate candy is associated with a 19.7481 increase in winpercent on average holding all other predictors fixed.

Std. Error:

- 95% CI for sugar-percent: [-0.2292 , 18.4088] (contains 0)
- 95% CI for chocolate: [11.9507 , 27.5455] (does not contain 0)

P-value:

- P-value is closely related to SE. 95% CI that don't contain 0 guarantee that p-value < 0.05 by definition

# LR Possible Improvements

Model improvements:

- Add an interaction term if you suspect combining two variables help explain the response.

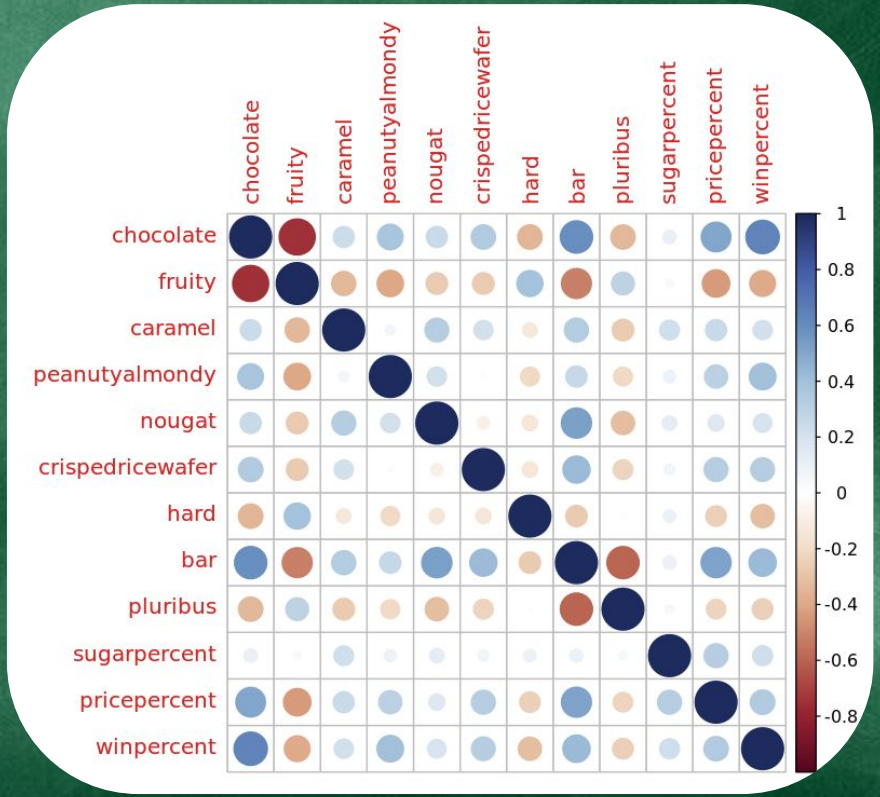
```
Call:
lm(formula = price ~ bedrooms * waterfront, data = house)

Residuals:
     Min       1Q   Median       3Q      Max
-3392480  -196842   -59682   109158  6857838

Coefficients:
                   Estimate Std. Error t value             Pr(>|t|)
(Intercept)          133202       8554  15.571 < 0.0000000000000002 ***
bedrooms             118160       2446  48.304 < 0.0000000000000002 ***
waterfront          -255355      84617  -3.018              0.00255 **
bedrooms:waterfront  422354      24375  17.327 < 0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 332800 on 21609 degrees of freedom
Multiple R-squared:  0.1785,    Adjusted R-squared:  0.1784
F-statistic:  1565 on 3 and 21609 DF,  p-value: < 0.00000000000000022
```

```
Call:
lm(formula = price ~ bedrooms + waterfront, data = house)

Residuals:
     Min       1Q   Median       3Q      Max
-3518509  -196152   -58689   109311  6846657

Coefficients:
            Estimate Std. Error t value             Pr(>|t|)
(Intercept)   118862       8573   13.87 <0.0000000000000002 ***
bedrooms      122414       2451   49.95 <0.0000000000000002 ***
waterfront   1138975      26344   43.23 <0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 335100 on 21610 degrees of freedom
Multiple R-squared:  0.1671,    Adjusted R-squared:  0.167
F-statistic:  2168 on 2 and 21610 DF,  p-value: < 0.00000000000000022
```

# LR Possible Improvements

Model improvements:

- Drop the predictors that are highly collinear

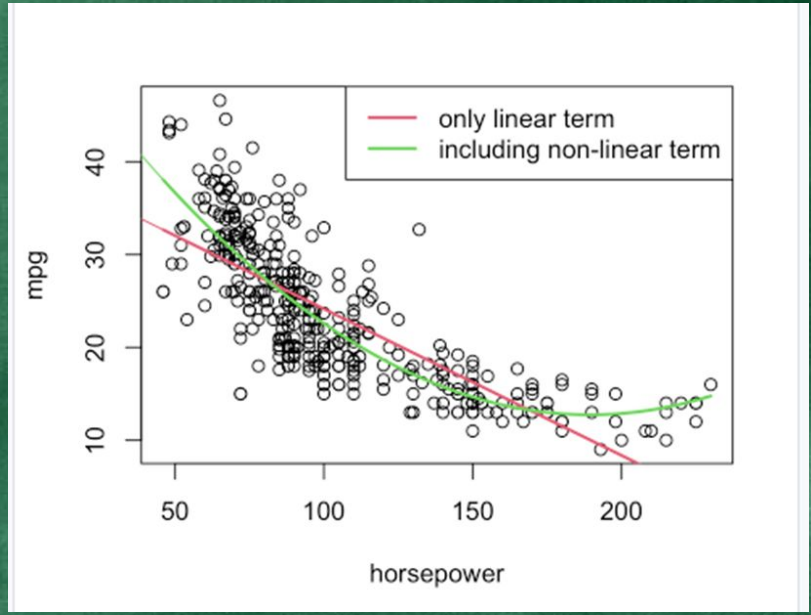



# LR Possible Improvements

Model improvements:

- Add non linear terms for variables with non-linear relation to response

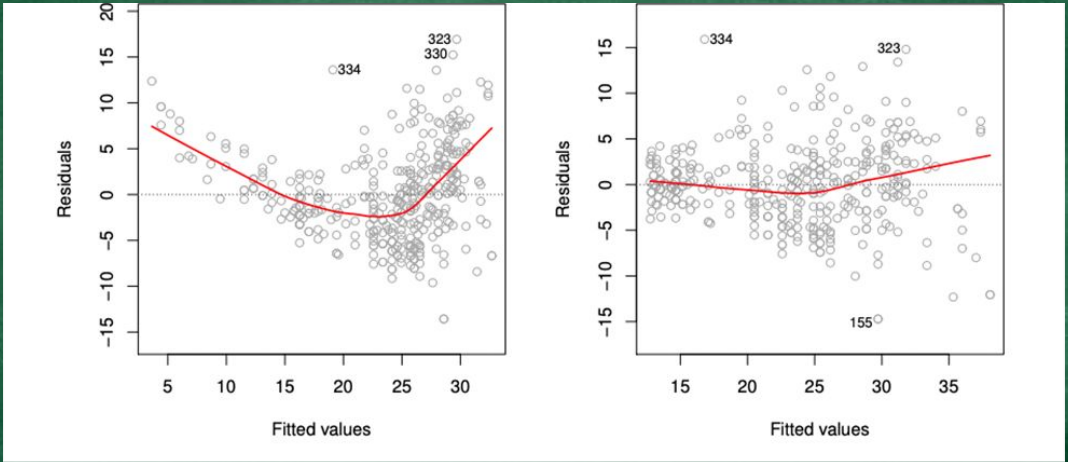
```
> lm_nonlinear <- lm(mpg~horsepower + I(horsepower ^ 2), Auto)
> summary(lm_nonlinear)

Call:
lm(formula = mpg ~ horsepower + I(horsepower^2), data = Auto)

Residuals:
     Min       1Q   Median       3Q      Max
-14.7135  -2.5943  -0.0859   2.2868  15.8961

Coefficients:
                  Estimate Std. Error t value            Pr(>|t|)
(Intercept)     56.9000997  1.8004268   31.60 <0.0000000000000002 ***
horsepower      -0.4661896  0.0311246  -14.98 <0.0000000000000002 ***
I(horsepower^2)  0.0012305  0.0001221   10.08 <0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.374 on 389 degrees of freedom
Multiple R-squared:  0.6876,    Adjusted R-squared:  0.686
F-statistic:   428 on 2 and 389 DF,  p-value: < 0.00000000000000022
```



# LR Possible Improvements

Model improvements:

- Read the residual plots
- A plot with obvious pattern would indicate non-linearity (left).
- Look to add non-linear term



# Classification: Logistic regression

- Binary response variable
- Predicting probability of response = 1

$$y = f(X) + \varepsilon$$

$$\text{Prob}(y=1) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_d x_d$$

Estimate

$$\text{Prob}(y=1) = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_d x_d$$

# Classification: Logistic regression

Probabilities have to be [0,1], so the equation for logistic regression must be modified

$$Prob(y = 1) = b_0 + b_1x_1 + b_2x_2 + \ldots + b_Dx_D$$

$$\downarrow$$

$$Prob(y = 1|X) = \frac{\exp(b_0 + b_1x_1 + b_2x_2 + \ldots + b_Dx_D)}{1 + \exp(b_0 + b_1x_1 + b_2x_2 + \ldots + b_Dx_D)}$$

$$\downarrow$$

$$\frac{p(X)}{1-p(X)} = \exp(b_0 + b_1x_1 + b_2x_2 + \ldots + b_Dx_D)$$

$$\downarrow$$

$$\log\left(\frac{p(X)}{1-p(X)}\right) = b_0 + b_1x_1 + b_2x_2 + \ldots + b_Dx_D$$

# Classification: Logistic regression

$$\log\left(\frac{p(X)}{1-p(X)}\right) = b_0 + b_1x_1 + b_2x_2 + \ldots + b_Dx_D$$

$\frac{p(X)}{1-p(X)}$ : odds

Every unit change in $x_n$ changes the log odds by $b_1$

$\log\left(\frac{p(X)}{1-p(X)}\right)$ : Log-odds (logit)

# Classification: Logistic regression

To estimate the $\beta_n$ we will use maximize Likelihood estimation

$$L(b_0, b_1, \dots, b_D) = \prod_{i:y_i=1} Prob(y_i = 1) \prod_{i:y_i=0} (1 - Prob(y_i = 1))$$

For the sample units that indeed have a response variable equal to 1, we want $Prob(y_i = 1)$ to be large

For the sample units that indeed have a response variable equal to 0, we want $Prob(y_i = 1)$ to be small, and thus $1 - Prob(y_i = 1)$ to be large

Maximize the product of: prob(y=1) and prob(y!=1) for all sample units y=1

# Classification: Linear Discriminant Analysis (LDA)

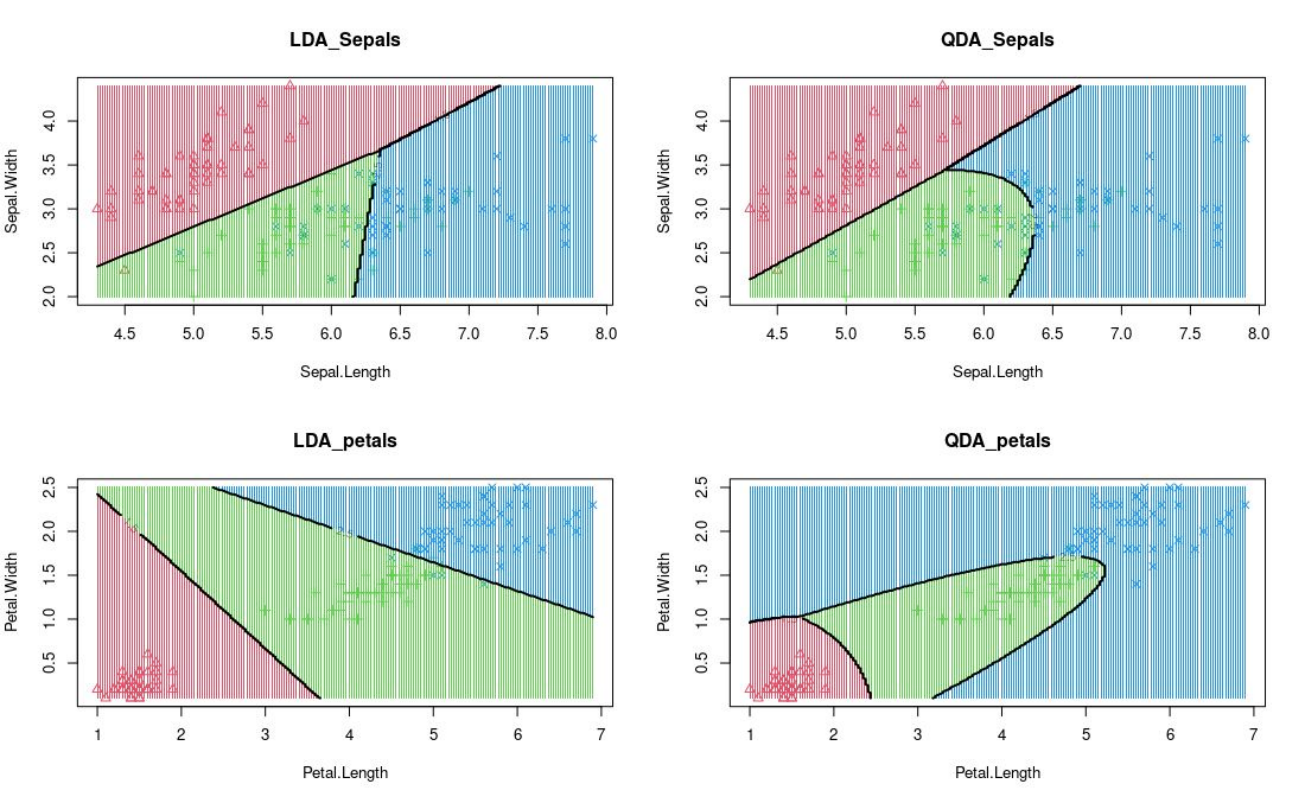
- Another Classification Method based on Bayes Theorem

$$\Pr(Y = k|X = x) = \frac{\Pr(X = x|Y = k)\Pr(Y = k)}{\sum_{l=1}^{K}\Pr(X = x|Y = l)\Pr(X = x)}$$

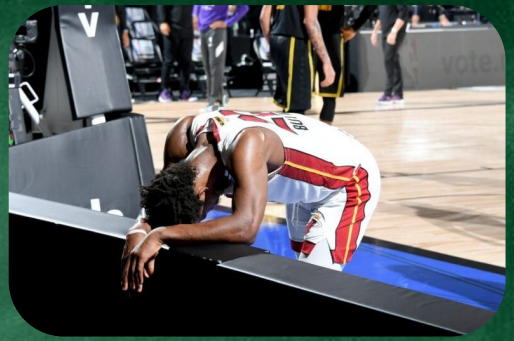
- Returns probabilities that a sample unit is in <u>each of k classes</u>
  - The math is messy

- Quadratic Discriminant Analysis (QDA) is an extension of LDA for quadratic classification
  - Better for complex patterns (Lower bias, higher variance)

# Classification: Linear Discriminant Analysis (LDA)




# Classification: Finally, Results

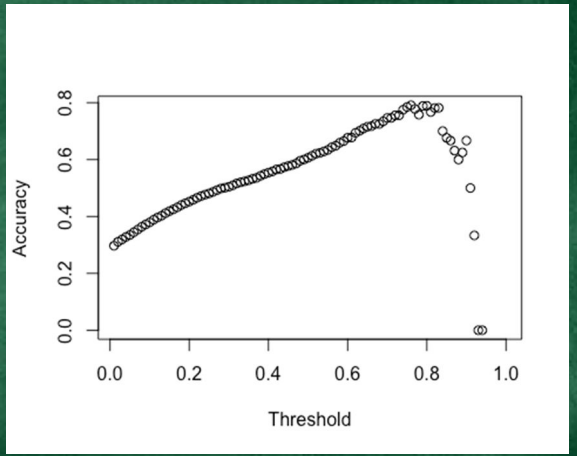


Now we have probabilities

Next step is to pick a threshold, that is

$$y_i = \begin{cases} 1, & if\ Prob(y_i = 1) > c \\ 0 & , \ otherwise \end{cases}$$

TO-DO: Try different C and see which performs *best*



# Classification: Confusion Matrix

| | | Predicted: Positive | Predicted: Negative |
|---|---|---|---|
| Actual | Positive | **True positive** | **False negative** |
| Actual | Negative | **False positive** | **True negative** |

# Classification: Metrics

| | | Predicted condition | | Sources: [4][5][6][7][8][9][10][11] view • talk • edit | |
|---|---|---|---|---|---|
| | Total population $= P + N$ | **Predicted positive** | **Predicted negative** | Informedness, bookmaker informedness (BM) $= TPR + TNR - 1$ | Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$ |
| **Actual condition** | **Positive (P)** [a] | **True positive** (TP), hit[b] | **False negative** (FN), miss, underestimation | True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$ | False negative rate (FNR), miss rate type II error [c] $= \frac{FN}{P} = 1 - TPR$ |
| | **Negative (N)**[d] | **False positive** (FP), false alarm, overestimation | **True negative** (TN), correct rejection[e] | False positive rate (FPR), probability of false alarm, fall-out type I error [f] $= \frac{FP}{N} = 1 - TNR$ | True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$ |
| | Prevalence $= \frac{P}{P + N}$ | Positive predictive value (PPV), precision $= \frac{TP}{TP + FP} = 1 - FDR$ | False omission rate (FOR) $= \frac{FN}{TN + FN} = 1 - NPV$ | Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$ | Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$ |
| | Accuracy (ACC) $= \frac{TP + TN}{P + N}$ | False discovery rate (FDR) $= \frac{FP}{TP + FP} = 1 - PPV$ | Negative predictive value (NPV) $= \frac{TN}{TN + FN} = 1 - FOR$ | Markedness (MK), deltaP (Δp) $= PPV + NPV - 1$ | Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$ |
| | Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$ | $F_1$ score $= \frac{2\,PPV \times TPR}{PPV + TPR} = \frac{2\,TP}{2\,TP + FP + FN}$ | Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$ | Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$ | Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$ |

# Classification: Sampling

Unbalanced data set:

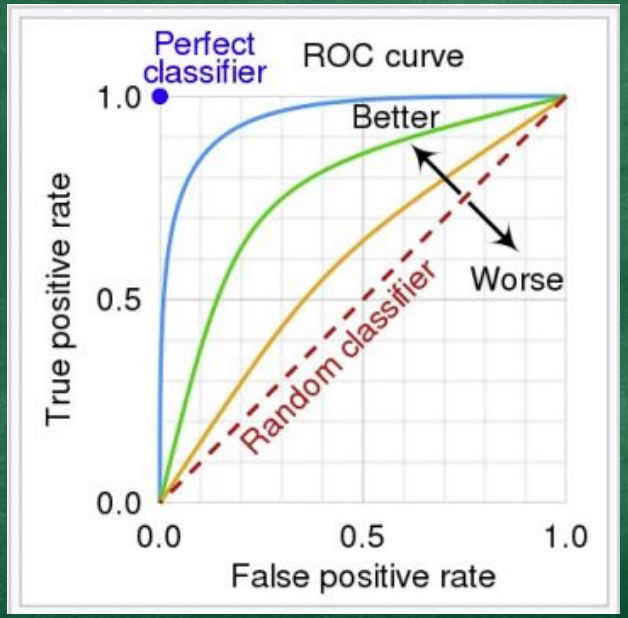
- Over or Under or Hybrid Sampling

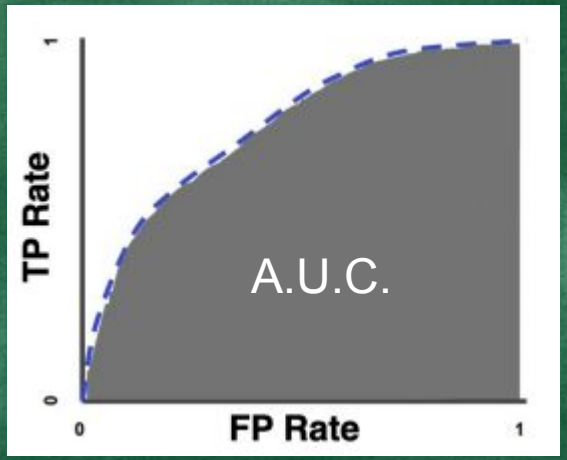
For any data set:

- Bootstrapping
  - Sample with replacement to create multiple data sets
- K-fold validation
  - Split data into k validation sets and average performance

# Classification: ROC Curve & AUC

A ROC (Receiver Operating Characteristics) curve is to measure the overall performance of a classification model under different thresholds.



# Classification: Logistic Results

Demo results are from a homework assignment looking at predicting good/bad credit (credit mix)

```
## 'data.frame':    54740 obs. of  8 variables:
##  $ Annual_Income           : num  72099 17444 15186 177697 19188 ...
##  $ Num_Credit_Card         : int  3 10 3 5 4 5 5 5 9 3 ...
##  $ Interest_Rate           : int  2 21 6 9 2 10 10 6 21 3 ...
##  $ Num_of_Loan             : num  4 5 1 1 3 5 3 -100 8 2 ...
##  $ Delay_from_due_date     : int  15 42 21 13 1 5 13 27 62 2 ...
##  $ Outstanding_Debt        : num  1422 2816.6 9.7 1182.9 464.1 ...
##  $ Monthly_Balance         : num  291 338 379 431 314 ...
##  $ Credit_Utilization_Ratio: num  36 35.5 36.6 35.1 28.9 ...
```

# Classification: Logistic Results

```
## Call:
## glm(formula = Credit_Mix ~ Annual_Income + Num_Credit_Card + Interest_Rate +
## Num_of_Loan + Delay_from_due_date + Outstanding_Debt + Monthly_Balance +
## Credit_Utilization_Ratio, family = binomial, data = trainingSet)
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)               6.937e-01  9.592e-02   7.232 4.77e-13 ***
## Annual_Income             4.627e-09  9.892e-09   0.468   0.6399
## Num_Credit_Card           8.013e-05  1.064e-04   0.753   0.4512
## Interest_Rate             1.736e-05  2.844e-05   0.610   0.5416
## Num_of_Loan              -4.357e-04  2.412e-04  -1.806   0.0709 .
## Delay_from_due_date      -1.159e-01  1.923e-03 -60.262  < 2e-16 ***
## Outstanding_Debt         -8.034e-04  2.476e-05 -32.441  < 2e-16 ***
## Monthly_Balance           1.382e-03  6.282e-05  21.993  < 2e-16 ***
## Credit_Utilization_Ratio  4.073e-03  2.850e-03   1.429   0.1529

## ---## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

# Classification: Logistic Results

```
## Delay_from_due_date       -1.159e-01  1.923e-03 -60.262  < 2e-16 ***
```

Delay_from_due_date: Each day that payment is delayed lowers log-odds of having good credit mix by .116 on average holding all other predictors fixed.

Log odds to odds:

- e^-.116 = odds ratio = 0.8907
- % change in odds = (1-odds ratio) = 0.1093

In terms of odds: The odds of having good credit mix decreases 11% on average, holding all other predictors fixed, for each day that payment is delayed.

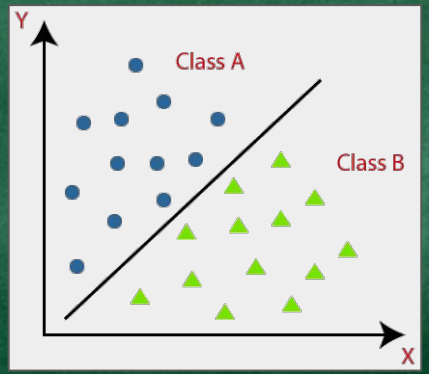
# Classification: LDA Results

```
## lda(Credit_Mix ~ Annual_Income + Interest_Rate + Outstanding_Debt +
##     Monthly_Balance + Credit_Utilization_Ratio, data = trainingSet)
##
## Prior probabilities of groups:
##         0       1
## 0.7562879 0.2437121
##
## Group means:
##   Annual_Income Interest_Rate Outstanding_Debt Monthly_Balance Credit_Utilization_Ratio
## 0      172446.0      75.80516       1643.0174      370.8372       32.07871
## 1      189063.3      76.45034        733.7616      503.5104       32.95380
## Coefficients of linear discriminants:
##                                    LD1
## Annual_Income             6.169737e-09
## Interest_Rate            2.730176e-05
## Outstanding_Debt          -7.042019e-04
## Monthly_Balance           2.321958e-03
## Credit_Utilization_Ratio  5.974218e-03
```

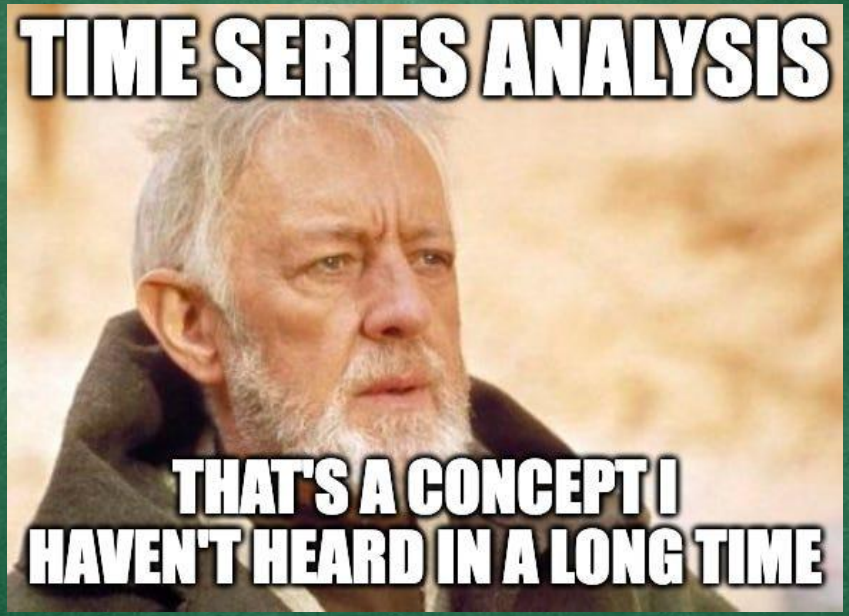
# Classification: LDA Results

These coefficients represent the linear combination of the predictor variables that best separate the two classes.

```
## Annual_Income              6.169737e-09
## Interest_Rate          2.730176e-05
## Outstanding_Debt           -7.042019e-04
## Monthly_Balance            2.321958e-03
## Credit_Utilization_Ratio   5.974218e-03
```

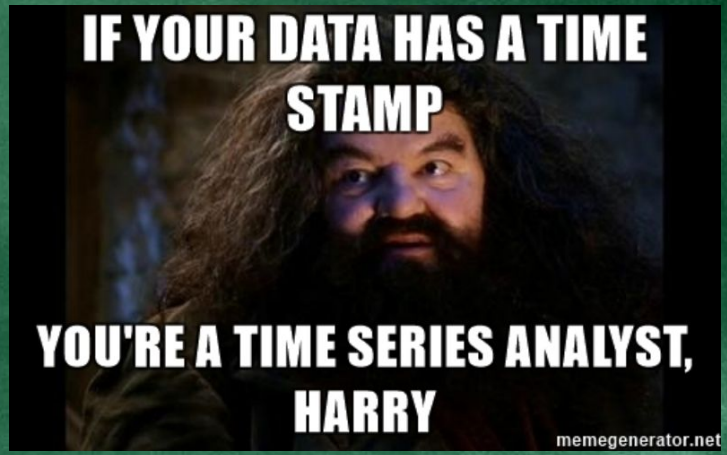


# Time Series


TIME SERIES ANALYSIS
THAT'S A CONCEPT I HAVEN'T HEARD IN A LONG TIME

# Time Series

- Sequence of data points recorded over time
- Time series data has an implicit order, meaning the sequence of data points matters
- N sample units with only ONE variable, but this variable is recorded on T different time stamps



# Time Series

- Involves machine learning algorithms to analyze and predict future values based on historical data collected over time, like stock prices or weather patterns

# Time Series Data: Components

- Trend: This represents the overall direction in which the data is moving over a long period. For example, the gradual increase in average global temperatures is a trend.

- Seasonality: These are patterns that repeat at regular intervals. For example, increased ice cream sales during summer months.

- Cyclic changes: These are fluctuations that occur irregularly but are not of fixed period, unlike seasonality. These could be tied to economic conditions like recessions.

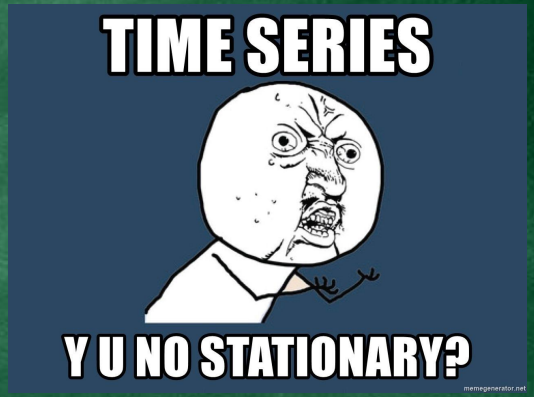
- Random noise: These are random fluctuations in the data that do not follow any pattern and are unpredictable.

# Time Series Analysis: Challenges

- Stationarity: Time series analysis often requires the data to be stationary. This means the statistical properties of the series like mean and variance should not be a function of time.

- High dimensionality: Some time series are multivariate and contain multiple variables at each time step, which increases the complexity of the model.

- Missing values and anomalies: Time series data can have gaps or outliers that need to be handled before analysis.



# Time Series Analysis: Models

## 1. Autoregressive (AR) Models

- predicts future behavior based on past behavior
- assumes that the current value of the series is a linear combination of previous values plus an error term. The model can be expressed as:

$$X_t = c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \ldots + \phi_p X_{t-p} + \epsilon_t$$

Where:

- $X_t$ is the current value of the series.
- $c$ is a constant.
- $\phi_1, \phi_2, \ldots, \phi_p$ are parameters of the model.
- $p$ is the order of the model (number of lagged observations).
- $\epsilon_t$ is white noise.

# Time Series Analysis: Models

## 2. Moving Average (MA) Models

- model uses past forecast errors in a regression-like model.
- Unlike AR models, which are based on the values of the series, MA models are based on the errors of past predictions. It can be expressed as:

$$X_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q}$$

Where:

- $\mu$ is the mean of the series.
- $\epsilon_t$ is the white noise at time $t$.
- $\theta_1, \theta_2, \ldots, \theta_q$ are the parameters of the model.
- $q$ is the order of the model.

# Time Series Analysis: Models

## 3. Autoregressive Moving Average (ARMA) Models

- combine both AR and MA models. They are used when a series is stationary, meaning it has constant mean and variance. The ARMA model is given by:

$$X_t = c + \phi_1 X_{t-1} + \ldots + \phi_p X_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \ldots + \theta_q \epsilon_{t-q}$$

- This model is specified by two parameters, p (order of the AR part) and q (order of the MA part)

# Time Series Analysis: Models

## 4. Autoregressive Integrated Moving Average (ARIMA) Models

- used to make a non-stationary series stationary before using ARMA models.
- done by differencing the series, i.e., computing the differences between consecutive observations.

The ARIMA model is generally denoted as ARIMA(p, d, q), where:

p is the number of autoregressive terms.

d is the number of differences needed for stationarity.

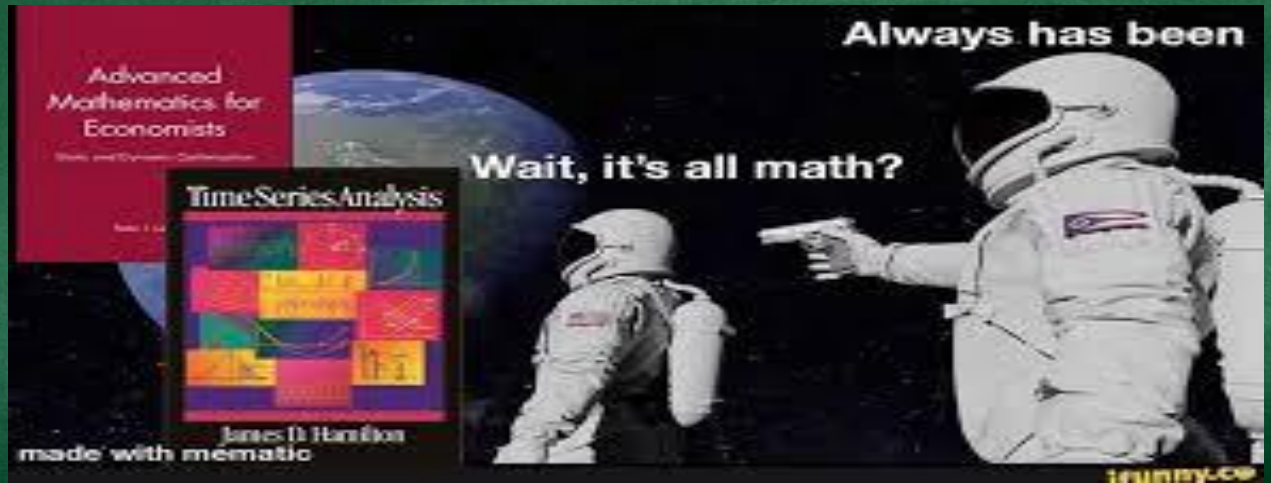
q is the number of moving average terms.

# Time Series Analysis: Models

The model can be written as:

$$\nabla^d X_t = c + \phi_1 \nabla^d X_{t-1} + \ldots + \phi_p \nabla^d X_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \ldots + \theta_q \epsilon_{t-q}$$

Here, $\nabla^d$ represents the differencing operator applied $d$ times to the series $X_t$.



# Variable Selection

Bias tends to be low as we have more predictors.

Variance tends to be low as we have less predictors.

In other words, when bias is already small enough, having less predictors might be able to reduce variance A LOT without hurting bias too much.

# Variable Selection: Best Subset

Try every combination of models with k variables from k = 1 to all variables

Not optimized.

# Variable Selection: Step-wise

Forward:

Step 0: start with a ***null*** model, a model that contains no predictors.

Step 1: For each of the D predictors, run a *simple* linear regression

$$y = b_0 + b_j\, x_j,$$

Find the regression with the lowest RSS.

Step 2: For each of the D-1 predictors, run linear regression:

$$y = b_0 + b_{j^*}\, x_{j^*} + b_j\, x_j,$$

where $x_{j^*}$ is the variable picked from Step 2. Find the regression with the lowest RSS.

Continue until the stopping rule is satisfied. A stopping rule could be you start to have variables with large p-values.

# Variable Selection: Step-wise

Backwards:

Step 0: start with a ***full*** model, a model that contains all the predictors.

Step 1: Remove the variable with the largest p-value.

Step 2: Run a linear regression with the rest of the D-1 variables. Remove the variable with the largest p-value.

Continue until a stopping rule is reached. For example, we can stop if the model only contains significant variables.

# Variable Selection: Metrics

All the selection methods (best subset, forward, backward) require a user-defined stopping rule. The easiest one to implement in R is to minimize the Akaike information criterion (AIC).

- In short, AIC is an estimator of the prediction error.

$$AIC = 2d + some\ training\ error$$

Why do we not want to use MSE or $R^2$?

- *Bias / Variance*

# Variable Selection: Shrinkage methods

Instead of purely minimizing the squared error, *ridge regression* minimizes

Loss + Penalty

$$\min (y - (\beta_0 + \beta_1 x_1 + \ldots + \beta_D x_D))^2 + \lambda(\beta_1{}^2 + \cdots + \beta_D{}^2),$$

In other words, in ridge regression, while we are still trying to find parameter estimates that could minimize MSE, a larger parameter estimate $\beta_j$ gets ***penalized***.

$\lambda$ is called a tuning parameter/hyperparameter. A larger $\lambda$ signals a heavier penalty on large parameter estimations.

# Variable Selection: Shrinkage methods

Similar to Ridge regression, the Lasso has the following *loss + penalty* objective function:

$$\min (y - (\beta_0 + \beta_1 x_1 + \ldots + \beta_D x_D))^2 + \lambda(|\beta_1| + \cdots + |\beta_D|)\,.$$

Optimal w/o penalty

Penalty

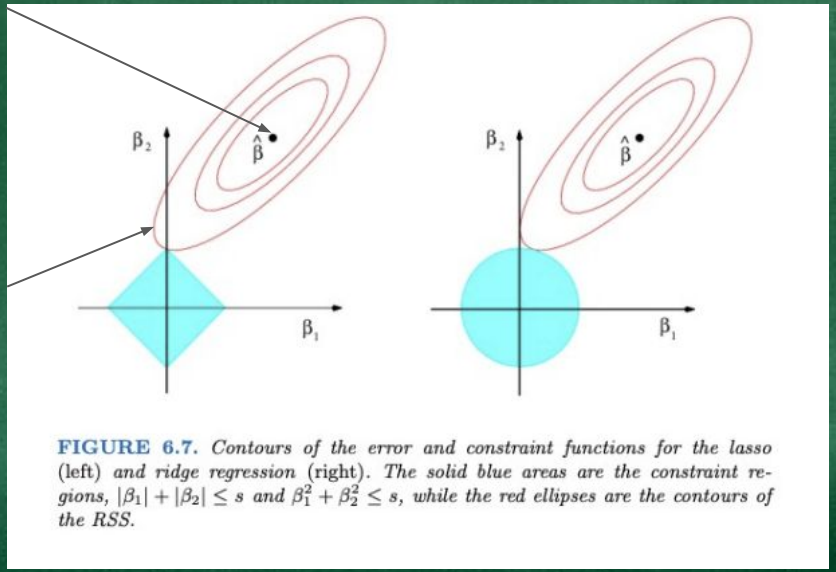



**FIGURE 6.7.** *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions,* $|\beta_1| + |\beta_2| \leq s$ *and* $\beta_1^2 + \beta_2^2 \leq s$, *while the red ellipses are the contours of the RSS.*

Note that ridge (right) does not touch the y-axis when it makes contact with the constraint

# Variable Selection: Shrinkage methods

Practical difference:

- The parameter estimates could be really small in Ridge, but they will NEVER be 0
- Thus, can't be eliminated from the model

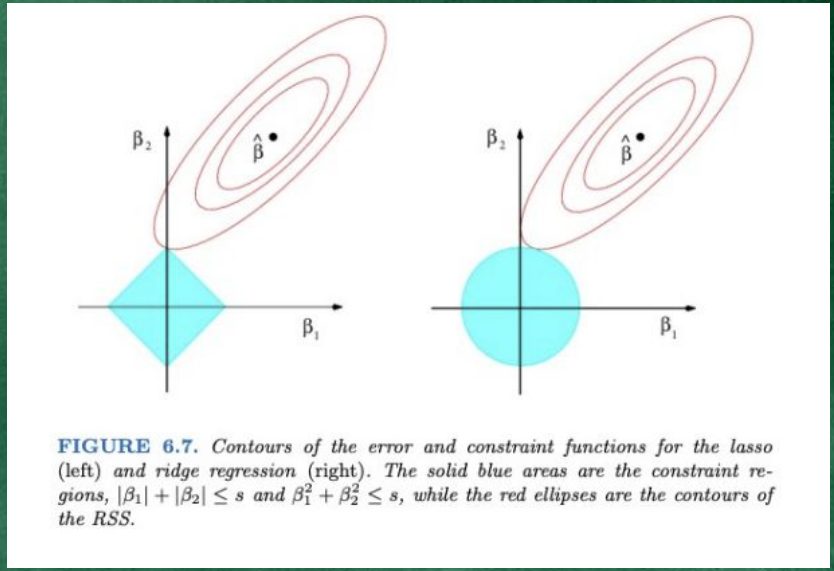
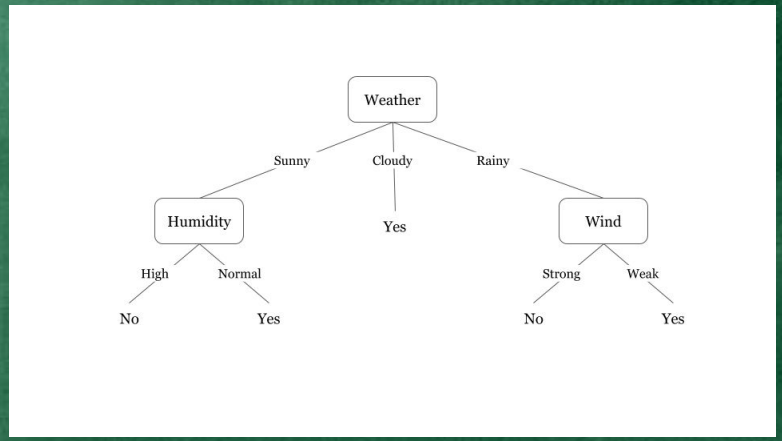



**FIGURE 6.7.** *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions,* $|\beta_1| + |\beta_2| \leq s$ *and* $\beta_1^2 + \beta_2^2 \leq s$, *while the red ellipses are the contours of the RSS.*

# Decision Trees

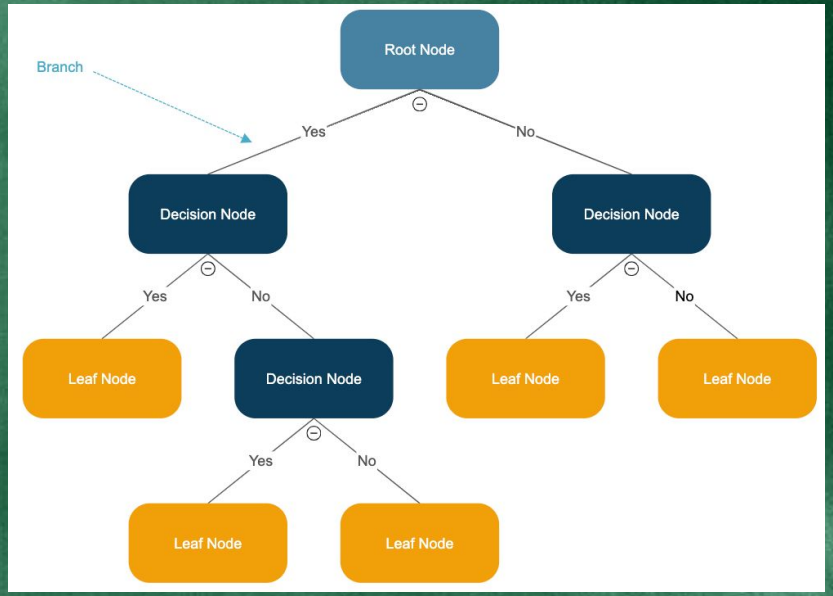
- Definition: A decision tree is a flowchart-like structure used for classification and regression tasks.




- Importance: Simple to understand, interpret, and visualize.
- Application: Used in finance, healthcare, marketing, and more.

# Components of a Decision Tree

- Root Node: Represents the entire dataset and splits into child nodes.
- Decision Nodes: Points where the dataset is split based on a feature.
- Leaf Nodes: Final output labels or values.



# Classification vs. Regression Trees

- Classification Trees: Used when the target variable is categorical (e.g., spam vs. not spam).
- Regression Trees: Used when the target variable is continuous (e.g., predicting house prices).

# Strengths and Weaknesses of Decision Trees

Advantages:

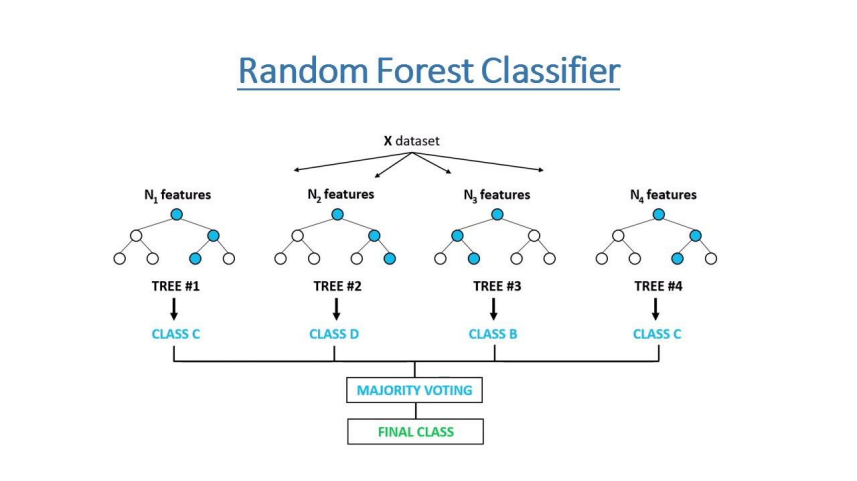
- Easy to interpret and visualize
- Requires minimal data preprocessing
- Handles both numerical and categorical data

Disadvantages:

- Prone to overfitting
- Can be unstable with small data changes
- Not as accurate as ensemble methods

# Understanding Random Forest Classifier

- Definition: A Random Forest is an ensemble learning method using multiple decision trees.
- Importance: Reduces overfitting, improves accuracy compared to single decision trees.
- Application: Used in finance, healthcare, and image recognition.



# How do random forest work

- Constructs multiple decision trees from randomly sampled data.
- Aggregates predictions (majority vote for classification, average for regression).
- Reduces variance and improves generalization.

# Using boosting to improve Decision Tree based models

**Definition:** Boosting is an ensemble learning technique that combines multiple weak models to create a strong predictive model.

**How it Works:** Models are trained sequentially, with each new model focusing on correcting the errors of the previous ones.

**Main Algorithms:**

- **AdaBoost:** Adjusts weights to focus on misclassified samples.
- **Gradient Boosting (GBM):** Uses gradient descent to minimize errors.
- **XGBoost:** Optimized GBM with speed and performance improvements.
- **LightGBM:** Faster training, handles large datasets efficiently.
- **CatBoost:** Specialized for categorical data.

# Using boosting to improve Decision Tree based models

**Advantages:**

- Reduces bias and variance for high accuracy.
- Works well with structured data.
- Handles both classification and regression tasks.

**Challenges:**

- Computationally intensive.
- Requires careful hyperparameter tuning to avoid overfitting.

# Strengths and Weaknesses of Decision Trees

Advantages:

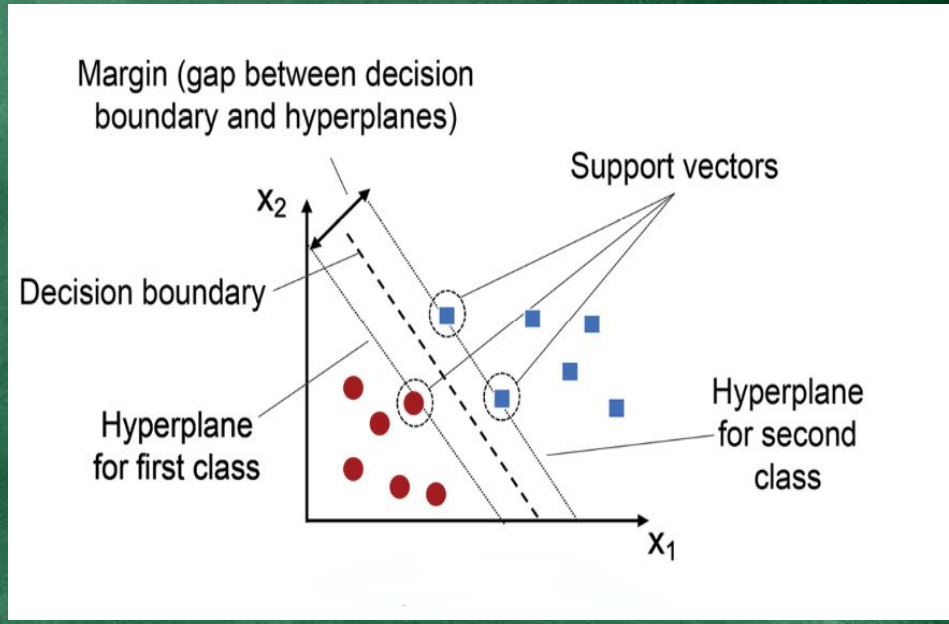
- Easy to interpret and visualize
- Requires minimal data preprocessing
- Handles both numerical and categorical data

Disadvantages:

- Prone to overfitting
- Can be unstable with small data changes
- Not as accurate as ensemble methods

# Support Vector Machines (SVM)

- Definition: SVM is a supervised learning algorithm used for classification and regression tasks.
- Importance: Effective in high-dimensional spaces, robust to overfitting.
- Application: Used in image recognition, bioinformatics, and text classification.

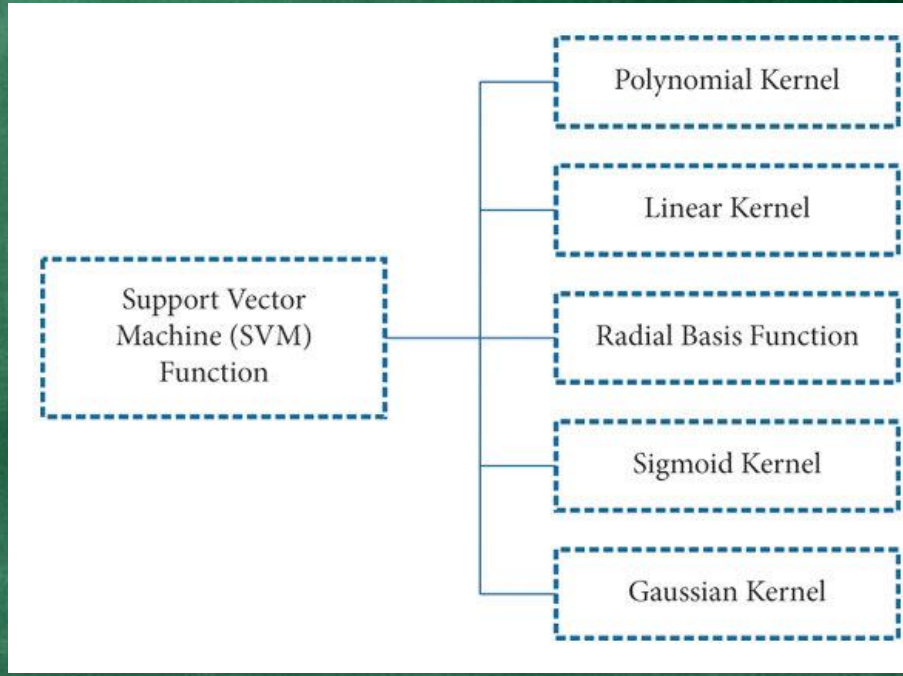


# Support Vector Machines (SVM)

- Identifies the optimal hyperplane that best separates different classes.
- Maximizes the margin between data points and the decision boundary.
- Uses support vectors to define the margin.

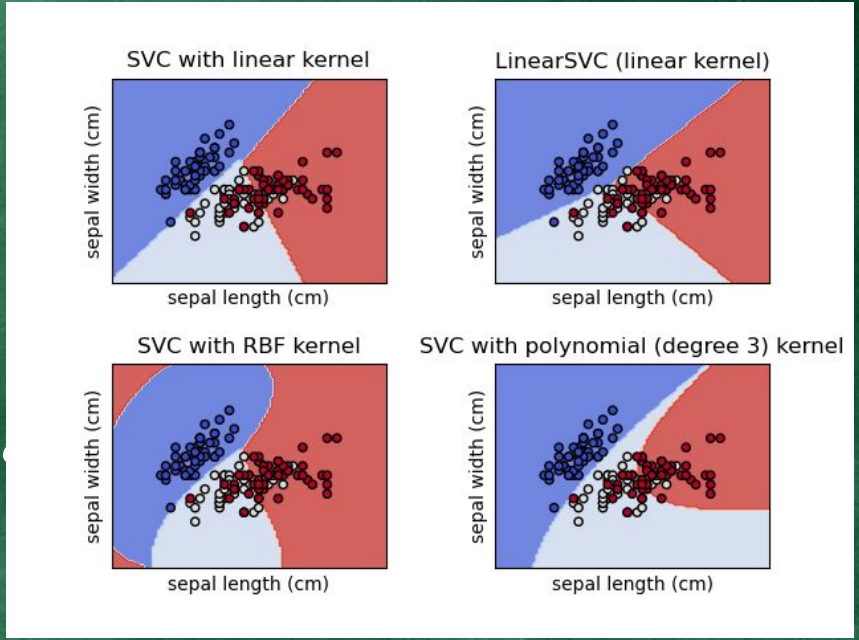
# Linear vs. Non Linear SVM

- Linear SVM: Works well when data is linearly separable.
- Non-Linear SVM: Uses kernel functions to map data into higher dimensions for better separation.



# Kernel Functions in SVM

- Linear Kernel: Suitable for linearly separable data.
- Polynomial Kernel: Captures complex relationships.
- Radial Basis Function (RBF) Kernel: Effective for non-linearly
- Sigmoid Kernel: Mimics neural networks.



# Pros and Cons of SVM

Advantages:

- Effective in high-dimensional spaces.
- Works well with small datasets.
- Robust to overfitting (especially with regularization).

Disadvantages:

- Computationally expensive for large datasets.
- Choice of kernel impacts performance.
- Difficult to interpret compared to decision trees.