Data partitioning and Evaluation of regression models

Yang Wang

College of Charleston

Prediction accuracy measures

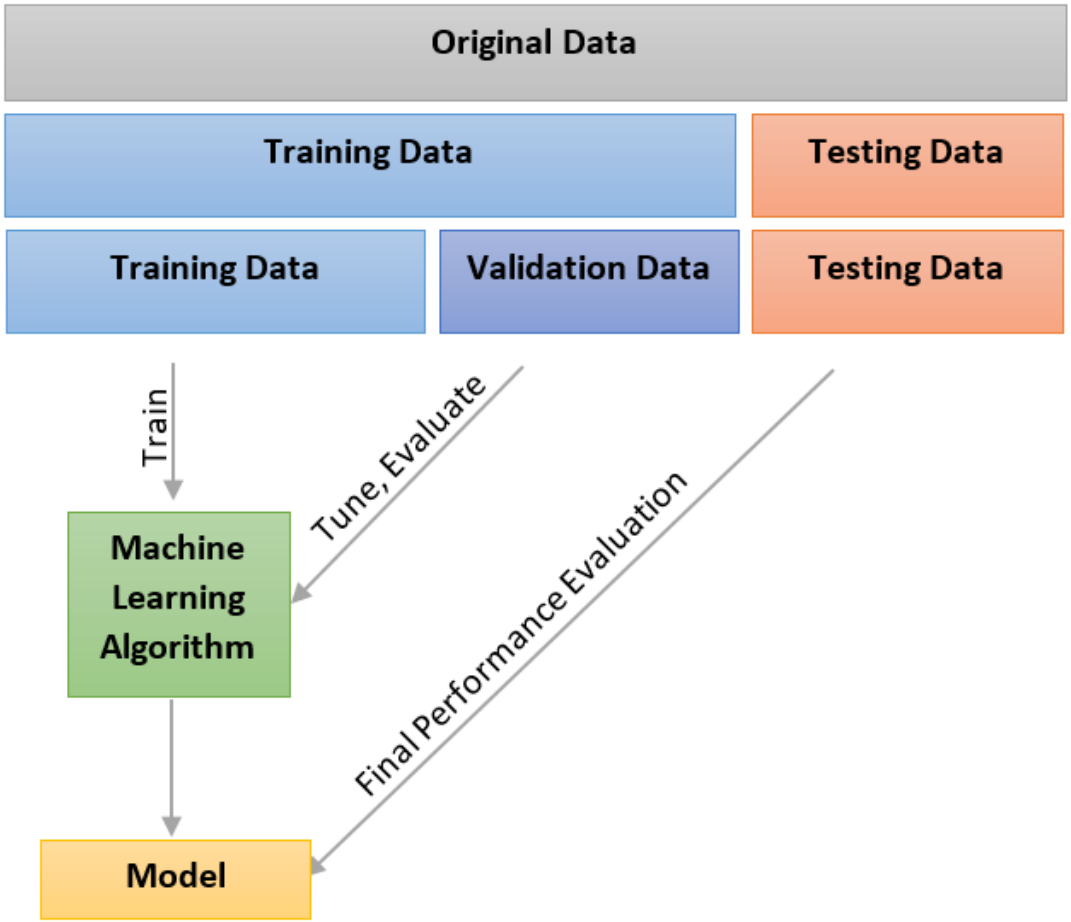
- y_i : response variable value of the i^th observation
- $\hat{y}_i$: predicted response value for the i^th observation
- $e_i = y_i - \hat{y}_i$: error for the i^th observation

Prediction accuracy measures

- **Mean Error** = $\frac{1}{n}\sum_{i=1}^{n} e_i$. This the average error. The disadvantage of this measure is that the negative errors cancel out the positive errors of the same magnitude. ($e_i = y_i - \hat{y}_i$)
- **MAE** (Mean Absolute Error) = $\frac{1}{n}\sum_{i=1}^{n} |e_i|$. This gives the magnitude of the average absolute error.
- **MPE** (Mean Percentage Error) = $100 \times \frac{1}{n}\sum_{i=1}^{n} e_i/y_i$. This gives the percentage score of how predictions deviate from the actual values (on average) taking into account the direction of error.
- **MAPE** (Mean Absolute Percentage Error) = $100 \times \frac{1}{n}\sum_{i=1}^{n} |e_i/y_i|$. This gives the percentage score of how predictions deviate from the actual values (on average).
- **RMSE** (Root Mean Square Error) = $\sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}$. This is the square root of the mean squared error. Note that, this is similar to the standard error of estimate in linear regression. The square root ensures that it has the same unit as the response variable.

Data partition

- Typically a model building process involves partitioning the data into three parts: training, testing and validation



Data partition

- Training data: this is typically the largest partition containing the data used for building the various models that will be examined. Typically the same training data will be used to develop multiple models.
- Validation data: this is used for comparing the predictive performance of models and choosing the best one. In some algorithms this set will be used in an automated fashion to tune and improve the model.
- Test data: this is used for assessing the performance of the chosen model with new data. This is also called holdout or evaluation partition.

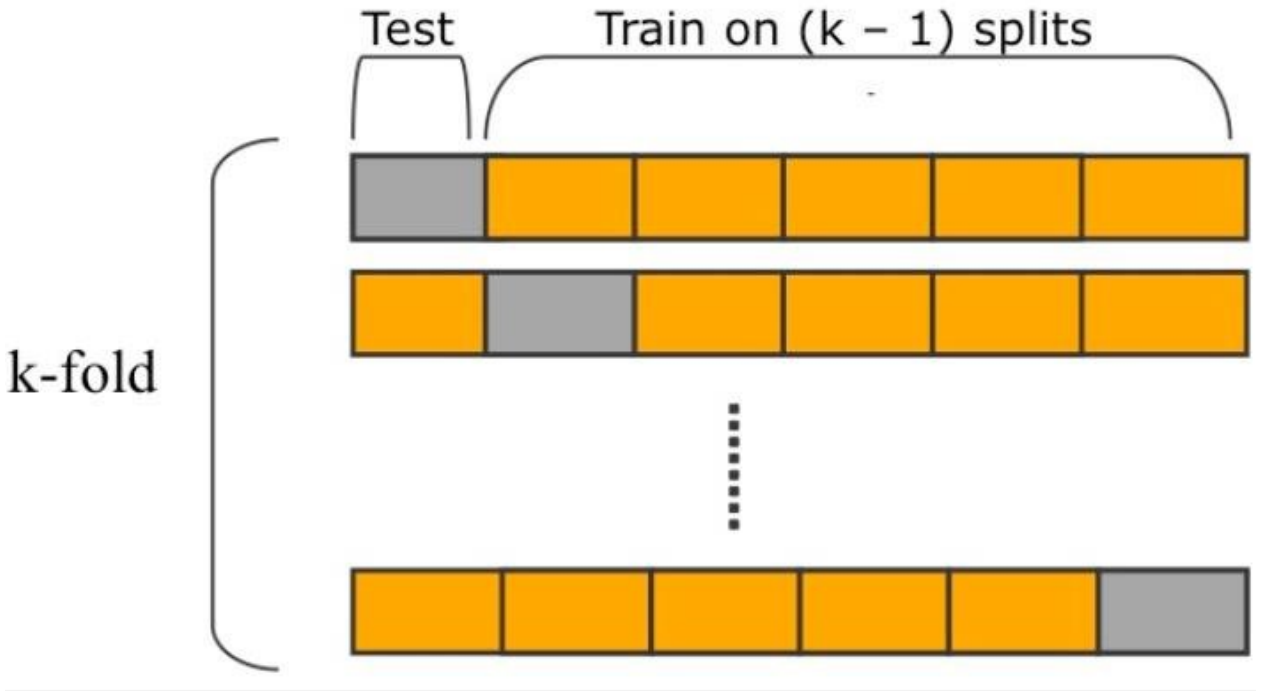
Data partition: Cross-validation

According to Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. ”The elements of statistical learning.” (2009), p.61)

- Combine the training and validation data to build the model using k-fold cross-validation or LOOCV.
- Then test the model on test data.

Data partition: Cross-validation

- Problem: different partitions will generate different models.
- Step 1: Partition the data into k "folds" or non-overlapping sub-samples.
- Step 2: Fit the model k times. Each time one of the folds is used as validation data and the remaining $k - 1$ folds are used as training data.
- Step 3: Combine the model's prediction from each fold (when they were used as validation data) for evaluating the performance of the model.



picture: https://www.researchgate.net/

Data partition: Leave-One-Out Cross-validation (LOOCV)

- For each i in $1, 2, ..., n$ do the following:
  - Step 1: Create the training data set with observations $1, \ldots, i-1, i+1, \ldots, n$.
  - Step 2: Fit the model on the training data and test it on the i^{th} observation.
- Combine the model's prediction from each step for evaluating the performance of the model.

Relative advantage and disadvantages of different approaches related to data partition

- Basic approach: random partitioning of the entire data set into two (training and test) or three (training, validation and test) groups followed by fitting the model using traing data (or training and validation data) and testing it on the test data:
  - Advantage: simplicity
  - Disadvantage: heavily dependent on specific partition
- Cross Validation:
  - Advantage: all observations in training and validation are used for model building
  - Disadvantage: LOOCV is computationally intensive
  - Disadvantage: K-fold: K must be chosen carefully, lower value of K leads to a biased model and a higher value of K can lead to variability in performance metrics of the model

Practice assignment

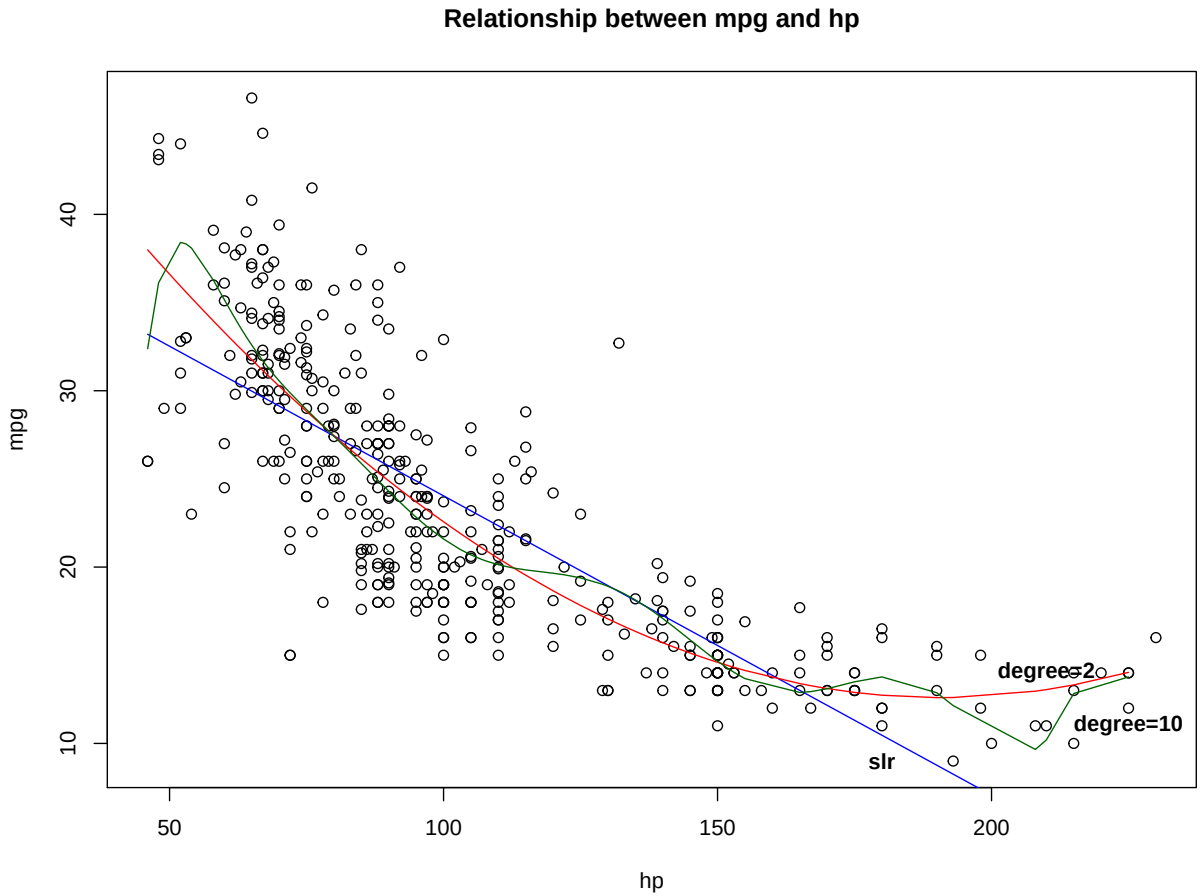
- Suppose the response variable is y and regressor is x. A polynomial regression model of degree q is given by: $y_i = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_q x^q + \epsilon_i$.
- Polynomials are extremely useful as they allow for more flexible models.
- Consider the autompg data set. The objective is to forecast mpg using hp. Create a 60:30:10 partition of the autompg data set.
- Fit polynomial regression models of 1 and higher degrees.
- Compare the MSE for each model in training and validation data.
- What degree seems to be an optimal fit?
- Fitting the optimal model on a combination of training and validation data sets
- Calculate the MSE for the fitted optimal model on the test data set.

Practice assignment

Note that, for this algorithm, the validation set is used as a part of the training process (to find the degree of the polynomial). As a result, a test set should be used for evaluating model performance. Thus, the algorithm has the following steps,

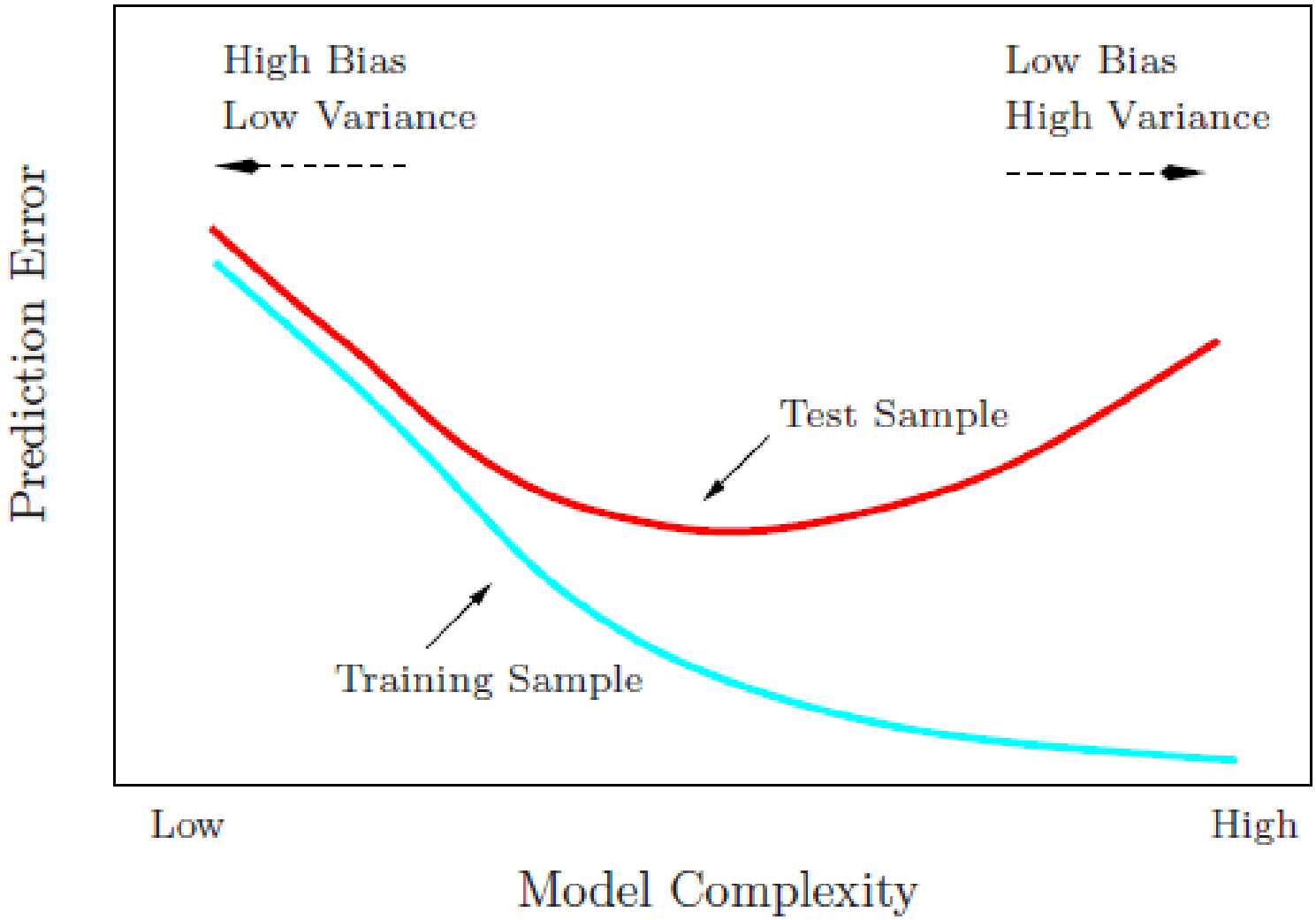
- partition data into training, validation, and test set.
- use training data for making predictions on validation data.
- use validation data for choosing the degree of the polynomial.
- use test for evaluating model performance with the chosen degree of the polynomial.

Bias Variance Trade-off



- Bias: how close the functional form of the predictive model is to the true relationship between the response and predictors
- Varince of the model: dependence of the predictive model on the data points that has been used for creating the model

Bias Variance Trade-off



Prediction accuracy measures for regression problem: Lift chart

Load ToyotaCorolla.csv
Partition data into training and test. Fit MLR model.

```
ToyotaCorolla <- ToyotaCorolla[, c(3:7)] # keep column 3,4,5,6,7

# use partition.2 function from myfunctions.R and create 60:40 partition
set.seed(0) ## set seed so that you get same partition each time
p2 <- partition.2(ToyotaCorolla, 0.6)
mydata.train <- p2$data.train
mydata.test <- p2$data.test

# replace missing values with median of the remaining values from the training data
summary(mydata.train$Weight)
med.wt.train <- median(mydata.train$Weight, na.rm = TRUE)
mydata.train$Weight[is.na(mydata.train$Weight)] <- med.wt.train
mydata.test$Weight[is.na(mydata.test$Weight)] <- med.wt.train

# run linear regression model
reg.m <- lm(Price ~ KM + factor(Fuel_Type) + HP + Weight, data = mydata.train)
summary(reg.m)
# prediction on training data
pred.t <- reg.m$fitted.values
# prediction on test data
pred.v <- predict(reg.m, newdata=as.data.frame(mydata.test))
```

Prediction accuracy measures for regression problem: Lift chart

- Objective: to find a set of records that has the highest predicted values
- Example: Find the cars with the highest selling price and market it to the target customers
- Rank order the data by predicted value
- Note that: in the absence of a predictive model, the best prediction about the response y is $\bar{y}$

```
> # Create Lift chart
> library(gains)
> gain <- gains(mydata.test$Price, pred.v) # creates the cumulative gain chart
> gain
```

Depth of File	N	Cume N	Mean Resp	Cume Mean Resp	Cume Pct of Total Resp	Lift Index	Cume Lift	Mean Model Score
10	57	57	18523.33	18523.33	17.3%	174	174	15922.04
20	57	114	12855.75	15689.54	29.3%	121	148	13152.92
30	58	172	10788.71	14036.94	39.6%	101	132	11846.52
40	57	229	10118.51	13061.61	49.0%	95	123	11240.88
50	58	287	9515.95	12345.06	58.0%	89	116	10693.91
60	57	344	9703.93	11907.43	67.1%	91	112	10240.58
70	57	401	9487.81	11563.50	76.0%	89	109	9784.99
80	58	459	8903.71	11227.40	84.4%	84	106	9160.45
90	57	516	8473.67	10923.21	92.3%	80	103	8294.90
100	58	574	8056.47	10633.54	100.0%	76	100	6572.46

Lift chart

- Data is sorted by predicted response and then partitioned into 10 groups.
- N denotes the size of each group.
- $CumeN[i] = CumeN[i-1] + N[i]$. For $i = 1$ consider $CumeN[i-1] = 0$.
- $MeanResp$ = average response for the corresponding group.
- $CumeMeanResp[i] = (CumeMeanResp[i-1] \times CumeN[i-1] + MeanResp[i] \times N[i])/CumeN[i]$
  Note that, $CumeMeanResp$ for the 10^{th} group = average of response values.
- $CumePctOfTotalResp[i]$ = ($CumeMeanResp[i] \times CumeN[i]$)/(sum of response values).
- $LiftIndex[i] = 100 \times MeanResp[i]$ / $CumeMeanResp$ for the 10^{th} group.
- $CumeLift[i] = 100 \times CumeMeanResp[i]$ / $CumeMeanResp$ for the 10^{th} group.
- $MeanModelScore$ = average predicted response for the corresponding group.

Depth of File	N	Cume N	Mean Resp	Cume Mean Resp	Cume Pct of Total Resp	Lift Index	Cume Lift	Mean Model Score
10	57	57	18523.33	18523.33	17.3%	174	174	15922.04
20	57	114	12855.75	15689.54	29.3%	121	148	13152.92
30	58	172	10788.71	14036.94	39.6%	101	132	11846.52
40	57	229	10118.51	13061.61	49.0%	95	123	11240.88
50	58	287	9515.95	12345.06	58.0%	89	116	10693.91
60	57	344	9703.93	11907.43	67.1%	91	112	10240.58
70	57	401	9487.81	11563.50	76.0%	89	109	9784.99
80	58	459	8903.71	11227.40	84.4%	84	106	9160.45
90	57	516	8473.67	10923.21	92.3%	80	103	8294.90
100	58	574	8056.47	10633.54	100.0%	76	100	6572.46

Prediction accuracy measures for regression problem: Lift chart

- Top 20% cases captures records with top 29.3% of total sales volume which is equivalent to a total of $114 \times 12855.75 = 1465556$ dollars.

```
# Plot Lift chart
x <- c(0, gain$depth)
pred.y <- c(0, gain$cume.pct.of.total)
avg.y <- c(0, gain$depth/100)
plot(x, pred.y, main = "Cumulative Lift Chart", xlab = "deciles",
     ylab = "Percent cumulative response", type = "l", col = "red")
lines(x, avg.y, type = "l"
```