Evaluating Supervised Machine Learning

Swati Mishra

Applications of Machine Learning (4AL3)

Fall 2024

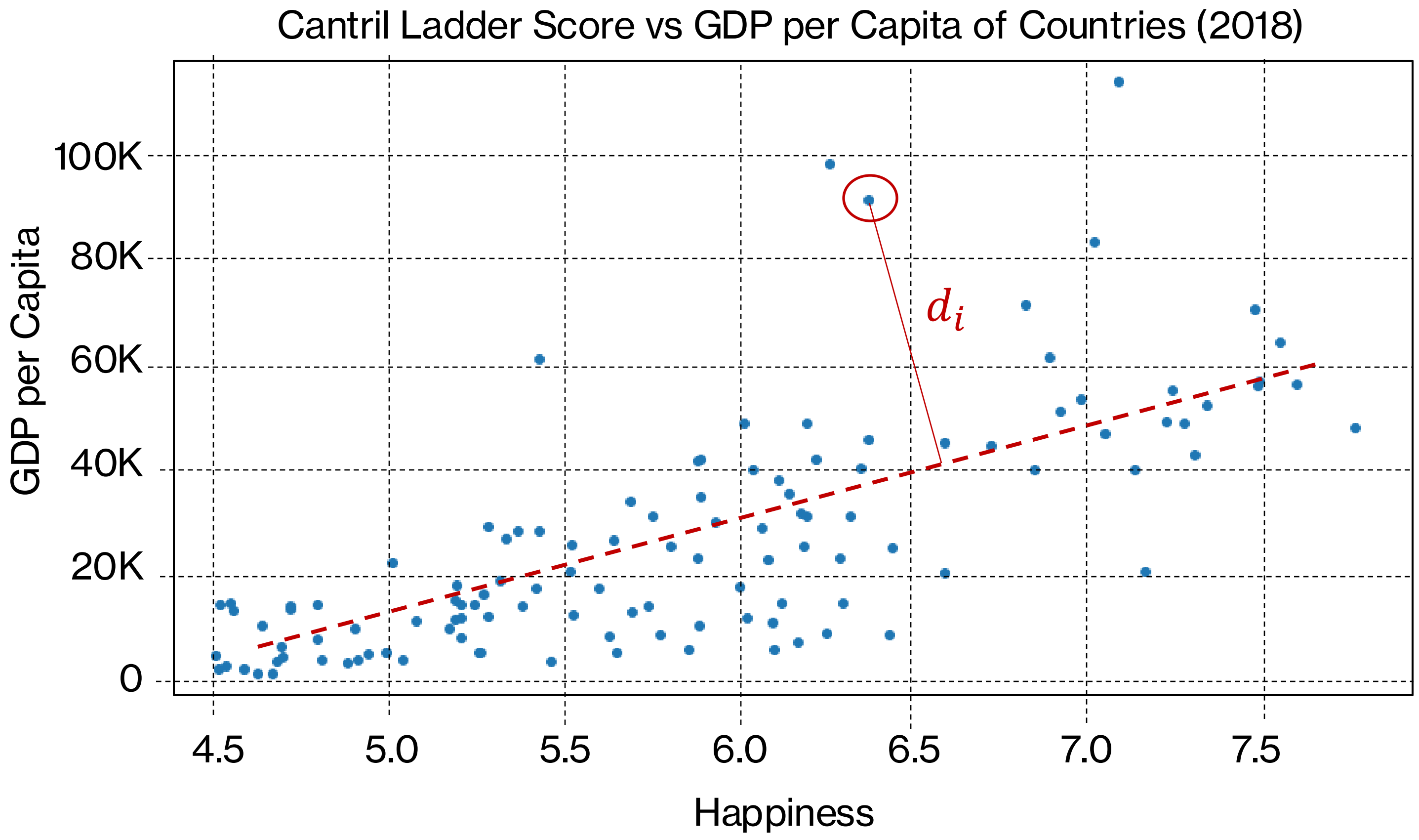
McMaster University

ENGINEERING

Review

- Fundamentals of Gradient Descent
- Implementing Gradient Descent
- Linear Regression – Gradient Descent
- Design Considerations (Learning Rate + Epochs)

Review Linear Regression



Step 2:

Hypothesize a linear model

$$y' = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \cdots + \beta_p * x_p$$

Step 3:

Select a Loss Function

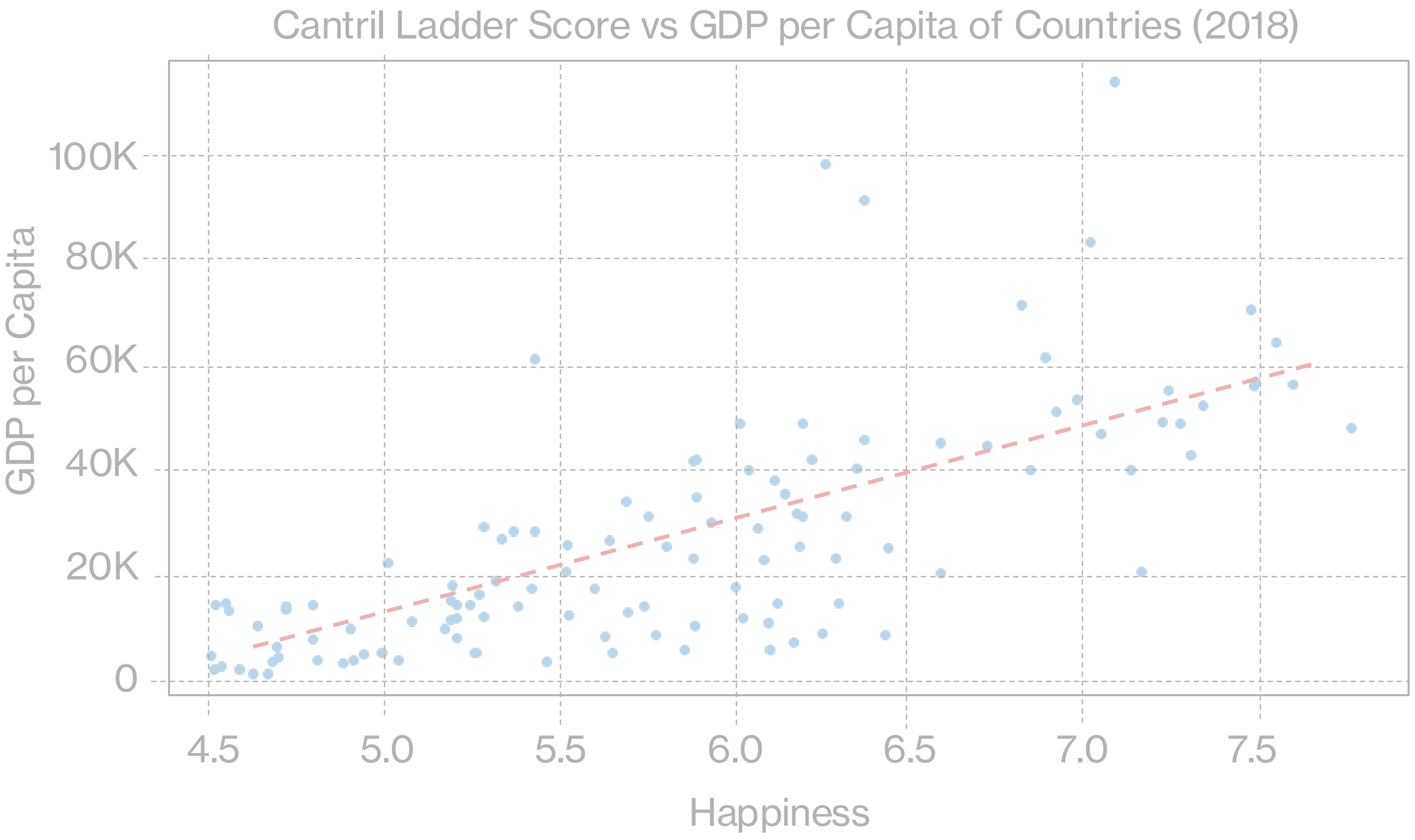
$$\sum_{i=0}^{n} {d_i}^2 \quad => \quad MSE = \frac{1}{n}\sum_{n=1}^{n}(y_i - y_i')^2$$

Step 4:

Find β such that it minimizes loss function

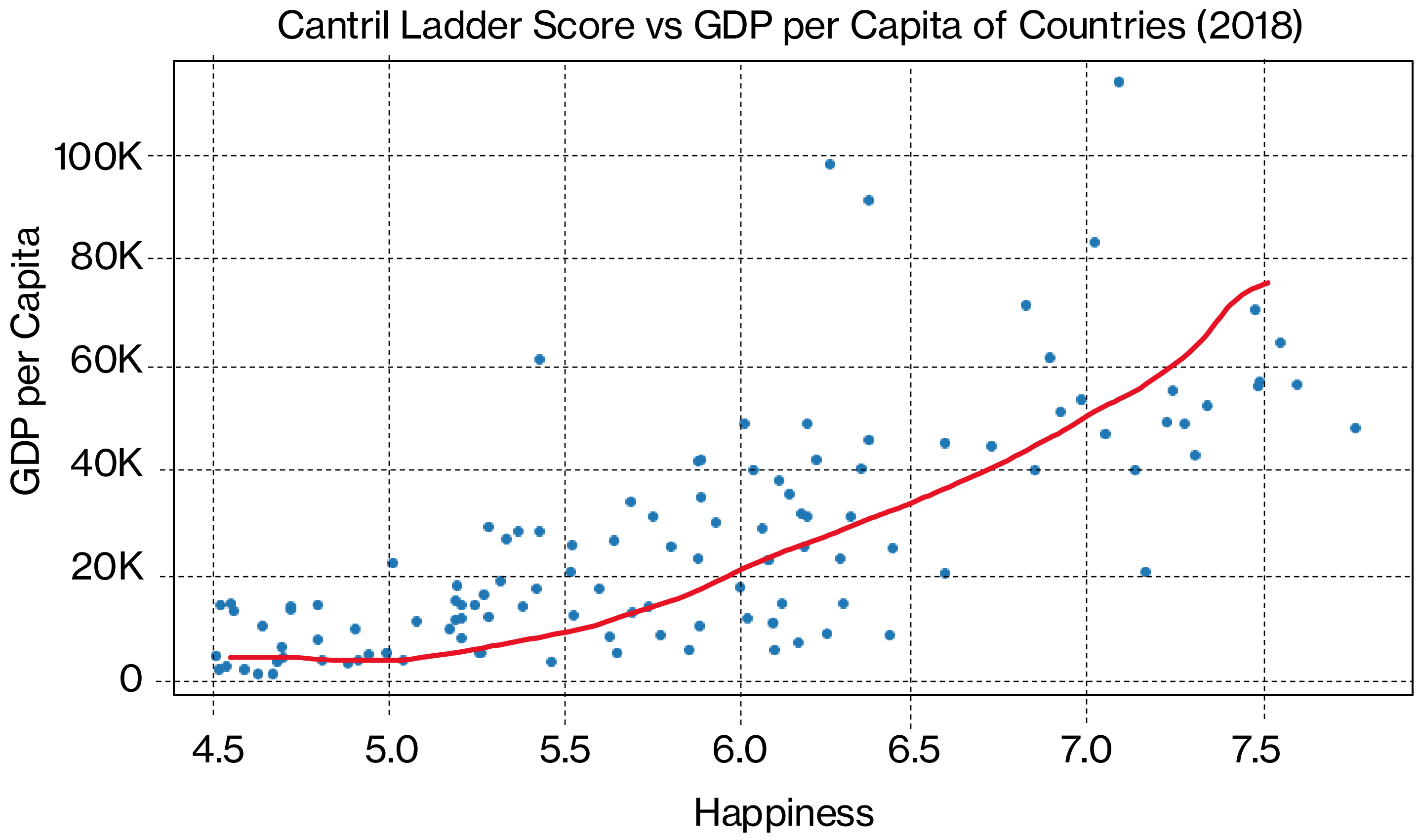
$$\beta' = (X^T X)^{-1} X^T y$$ or $$\frac{2}{n}(\beta . x - y) \cdot \boldsymbol{x^T}$$

Review Linear Regression



Not all relationships are linear

Review Linear Regression



A better fit!

Polynomial Regression

In polynomial regression, we assume that there is a non-linear relationship between its variables.

Our linear function:

$$y' = \beta_0 + \beta_1 * \mathrm{x}_1 + \epsilon$$

Our non-linear or n-degree polynomial function:

$$y' = \beta_0 + \beta_1 * \mathrm{x}_1 + \beta_2 * {\mathrm{x}_1}^2 + \beta_3 * {\mathrm{x}_1}^3 + \cdots + \beta_n * {\mathrm{x}_1}^n + \epsilon$$

Polynomial Regression

In polynomial regression, we assume that there is a non-linear relationship between its variables.

Our linear function:

$$y' = \beta_0 + \beta_1 * \mathrm{x}_1 + \epsilon$$

Our non-linear or n-degree polynomial function:

$$y' = \beta_0 + \beta_1 * \mathrm{x}_1 + \beta_2 * {\mathrm{x}_1}^2 + \beta_3 * {\mathrm{x}_1}^3 + \cdots + \beta_n * {\mathrm{x}_1}^n + \epsilon$$

This is different from multiple linear regression : $y' = \beta_0 + \beta_1 * \mathrm{x}_1 + \beta_2 * \mathrm{x}_2 + \cdots + \beta_p * \mathrm{x}_p$

McMaster University

Polynomial Regression

In polynomial regression, we assume that there is a non-linear relationship between its variables.

Our linear function:

$$y' = \beta_0 + \beta_1 * \mathrm{x}_1 + \epsilon$$

Our non-linear or n-degree polynomial function:

$$y' = \beta_0 + \beta_1 * \mathrm{x}_1 + \beta_2 * \mathrm{x}_1{}^2 + \beta_3 * \mathrm{x}_1{}^3 + \cdots + \beta_n * \mathrm{x}_1{}^n + \epsilon$$

This is different from multiple linear regression : $y' = \beta_0 + \beta_1 * \mathrm{x}_1 + \beta_2 * \mathrm{x}_2 + \cdots + \beta_p * \mathrm{x}_p$

Polynomial Regression

How to learn parameters for a polynomial regression model:

1. Hypothesize a non-linear model: $y' = \beta_0 + \beta_1 * \mathrm{x}_1 + \beta_2 * \mathrm{x}_1{}^2 + \beta_3 * \mathrm{x}_1{}^3 + \cdots + \beta_n * \mathrm{x}_1{}^n + \epsilon$

2. Select a Loss Function $MSE = \frac{1}{n}\sum_{n=1}^{n}(y_i - y_i{}')^2$

3. Find β such that it minimizes loss function

Polynomial Regression

How to learn parameters for a polynomial regression model:

1. Hypothesize a non-linear model: $y' = \beta_0 + \beta_1 * \mathrm{x}_1 + \beta_2 * {\mathrm{x}_1}^2 + \beta_3 * {\mathrm{x}_1}^3 + \cdots + \beta_n * {\mathrm{x}_1}^n + \epsilon$

2. Select a Loss Function $MSE = \frac{1}{n}\sum_{n=1}^{n}(y_i - y_i')^2$

3. Find β such that it minimizes loss function $\beta' = (\mathrm{X}^T X)^{-1}\mathrm{X}^T y$

Polynomial Regression

How to learn parameters for a polynomial regression model:

1. Hypothesize a non-linear model: $y' = \beta_0 + \beta_1 * \mathrm{x}_1 + \beta_2 * \mathrm{x}_1{}^2 + \beta_3 * \mathrm{x}_1{}^3 + \cdots + \beta_n * \mathrm{x}_1{}^n + \epsilon$

2. Select a Loss Function $MSE = \frac{1}{n}\sum_{n=1}^{n}(y_i - y_i')^2$

3. Find β such that it minimizes loss function $\beta' = (\mathrm{X}^T X)^{-1}\mathrm{X}^T y$

How can we implement polynomial regression?

Polynomial Regression

$$y' = \beta_0 + \beta_1 * \mathrm{x}_1 + \beta_2 * \mathrm{x}_1{}^2 + \beta_3 * \mathrm{x}_1{}^3 + \cdots + \beta_n * \mathrm{x}_1{}^n + \epsilon$$

- Above β can be easily estimated using OLS because this is just a standard linear model with predictors x_p , $\mathrm{x}_\mathrm{p}{}^2$, $\mathrm{x}_\mathrm{p}{}^3$.... $\mathrm{x}_p{}^n$
- To compute β'

$$\beta' = (\mathrm{X}^T X)^{-1} \mathrm{X}^T y$$

Polynomial Regression

$$y' = \beta_0 + \beta_1 * \mathrm{x}_1 + \beta_2 * \mathrm{x}_1{}^2 + \beta_3 * \mathrm{x}_1{}^3 + \cdots + \beta_n * \mathrm{x}_1{}^n + \epsilon$$

- Above β can be easily estimated using OLS because this is just a standard linear model with predictors x_p , $\mathrm{x}_\mathrm{p}{}^2$, $\mathrm{x}_\mathrm{p}{}^3$.... $\mathrm{x}_p{}^n$
- To compute β'

$$\beta' = (\mathrm{X}^T X)^{-1} \mathrm{X}^T y$$

$$X = \begin{vmatrix} x_1 & \mathrm{x}_1{}^2 \ldots & \mathrm{x}_1{}^{\mathrm{n}} \\ x_2 & \mathrm{x}_2{}^2 \ldots & \mathrm{x}_2{}^{\mathrm{n}} \\ \vdots & \vdots \ \ldots & \vdots \\ x_p & \mathrm{x}_p{}^2 \ldots & \mathrm{x}_p{}^{\mathrm{n}} \end{vmatrix}$$

$$y = \begin{vmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{vmatrix}$$

p = number of observations
n = degree of polynomial

Polynomial Regression

$$y' = \beta_0 + \beta_1 * x_1 + \beta_2 * x_1^2 + \beta_3 * x_1^3 + \cdots + \beta_n * x_1^n + \epsilon$$

- Above β can be easily estimated using OLS because this is just a standard linear model with predictors x_p , x_p^2, x_p^3.... x_p^n
- To compute β'

$$\beta' = (X^T X)^{-1} X^T y$$

Polynomial Regression

$$X = \begin{vmatrix} x_1 & x_1^2 \dots & x_1^n \\ x_2 & x_2^2 \dots & x_2^n \\ \vdots & \vdots \; \dots & \vdots \\ x_p & x_p^2 \dots & x_p^n \end{vmatrix} \qquad y = \begin{vmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{vmatrix}$$

p = number of observations

n = degree of polynomial

Multiple Linear Regression

$$X = \begin{vmatrix} x_{11} & x_{12} \dots & x_{1n} \\ x_{21} & x_{22} \dots & x_{2n} \\ \vdots & \vdots \; \dots & \vdots \\ x_{p1} & x_{p2} \dots & x_{pn} \end{vmatrix} \qquad y = \begin{vmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{vmatrix}$$

p = number of observations

n = number of features

Polynomial Regression - Implementation

$$y' = \beta_0 + \beta_1 * x_1 + \beta_2 * x_1{}^2 + \beta_3 * x_1{}^3 + \cdots + \beta_n * x_1{}^n + \epsilon$$

- Above β can be easily estimated using OLS because this is just a standard linear model with predictors x_p , $x_p{}^2$, $x_p{}^3$.... $x_p{}^n$
- To compute β'

$$\beta' = (X^T X)^{-1} X^T y$$

Polynomial Regression

$$X = \begin{vmatrix} x_1 & x_1{}^2 \ldots & x_1{}^n \\ x_2 & x_2{}^2 \ldots & x_2{}^n \\ \vdots & \vdots \ \ldots & \vdots \\ x_p & x_p{}^2 \ldots & x_p{}^n \end{vmatrix} \qquad y = \begin{vmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{vmatrix}$$

```
def __init__(self,x_:list,y_:list) -> None:

    self.input = np.array(x_)
    self.target = np.array(y_)
```

```
#arrange in matrix format
Y = (np.array([self.target])).T
```

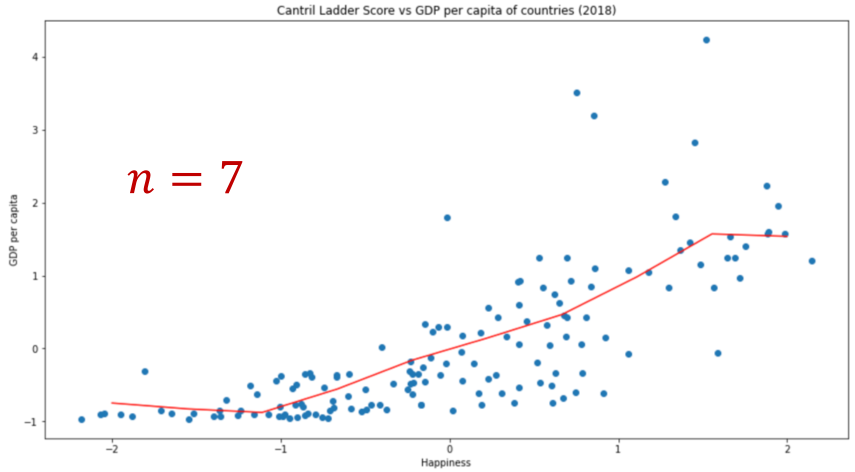
p = number of observations
n = degree of polynomial

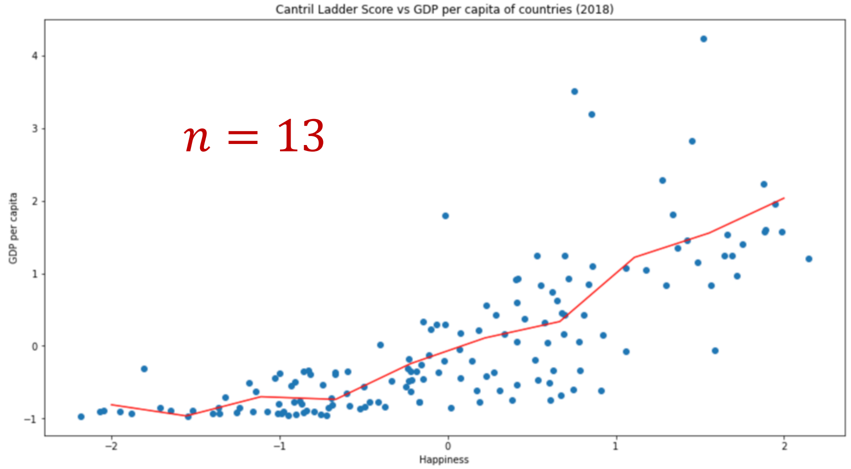
```
#arrange in matrix format
X = np.column_stack([self.input,self.input**2,self.input**3])
```

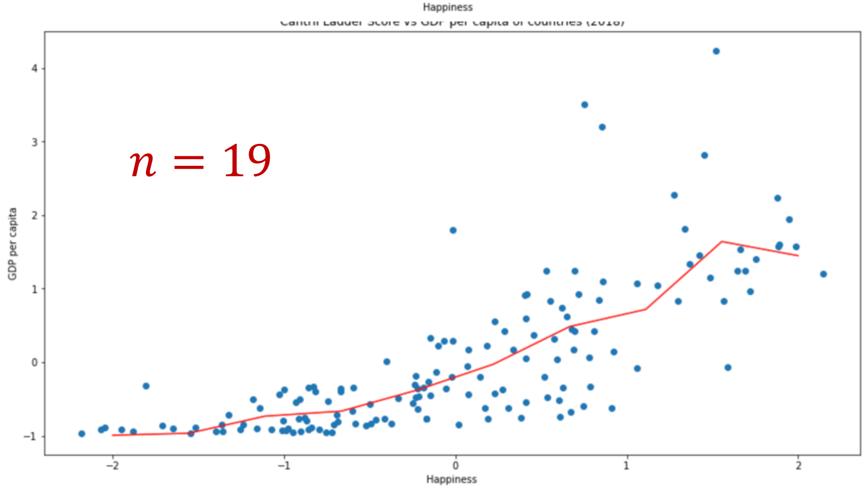
Polynomial Regression - Results

$$y' = \beta_0 + \beta_1 * x_1 + \beta_2 * {x_1}^2 + \beta_3 * {x_1}^3 + \cdots + \beta_n * {x_1}^n + \epsilon$$

n= degree of polynomial





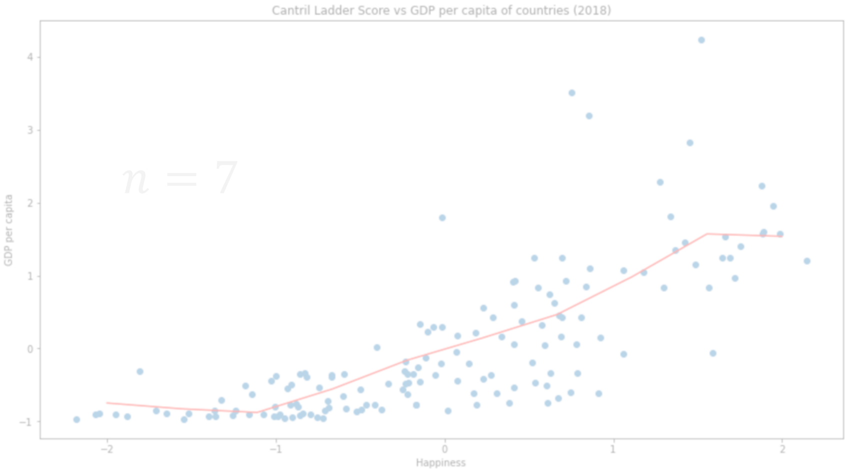


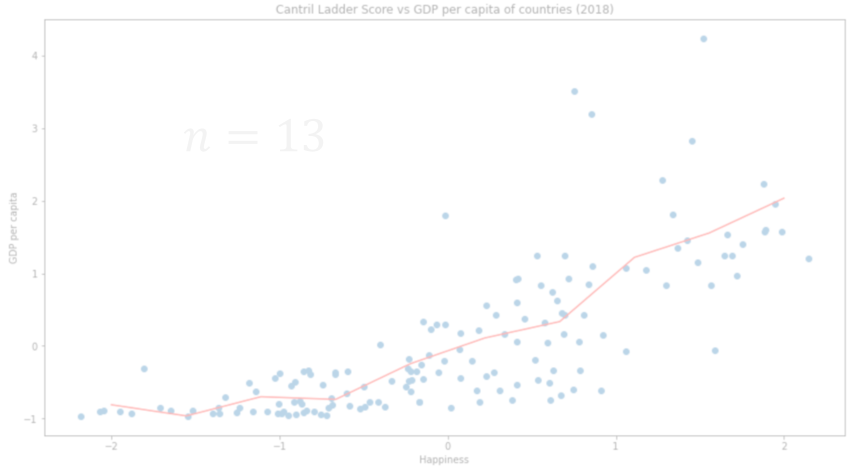
Polynomial Regression - Results

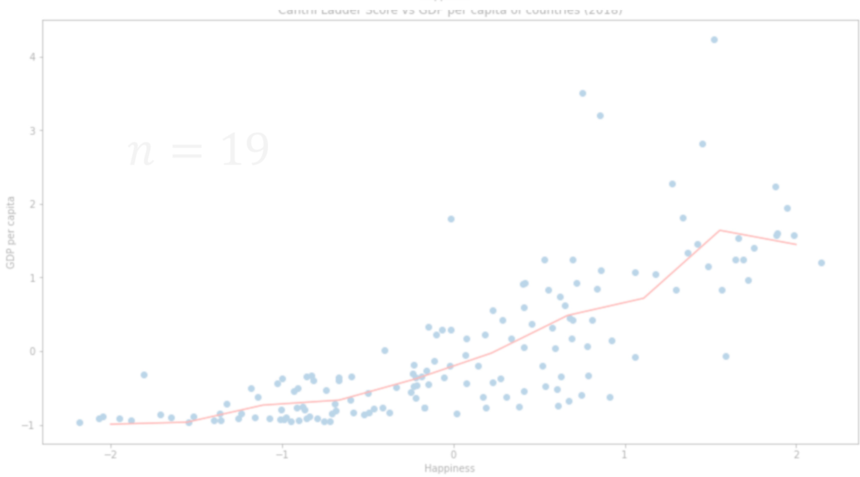
$$y' = \beta_0 + \beta_1 * x_1 + \beta_2 * {x_1}^2 + \beta_3 * {x_1}^3 + \cdots + \beta_n * {x_1}^n + \epsilon$$

What is a good model?

n= degree of polynomial







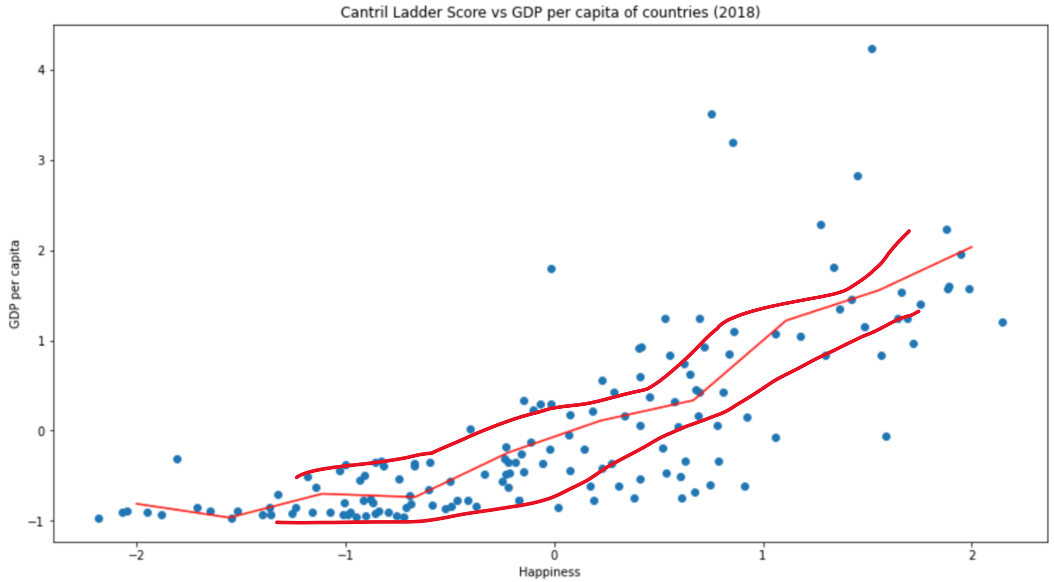
Evaluation

- To evaluate a model, we need to quantify how close the predicted response value is to the true response value for that observation.
- For regression, this difference is called **error** and should be very small for **that** observation
- But we don't care so much about the training data, we care about the accuracy on the **unseen** data instances.
- For regression, the MSE should be very small for **that** observation
- If we had a large number of test observations for regression model, we compute **test MSE.**

$$\text{Test } MSE = \frac{1}{n}\sum_{n=1}^{n}(y_i - y_i')^2$$

Evaluation

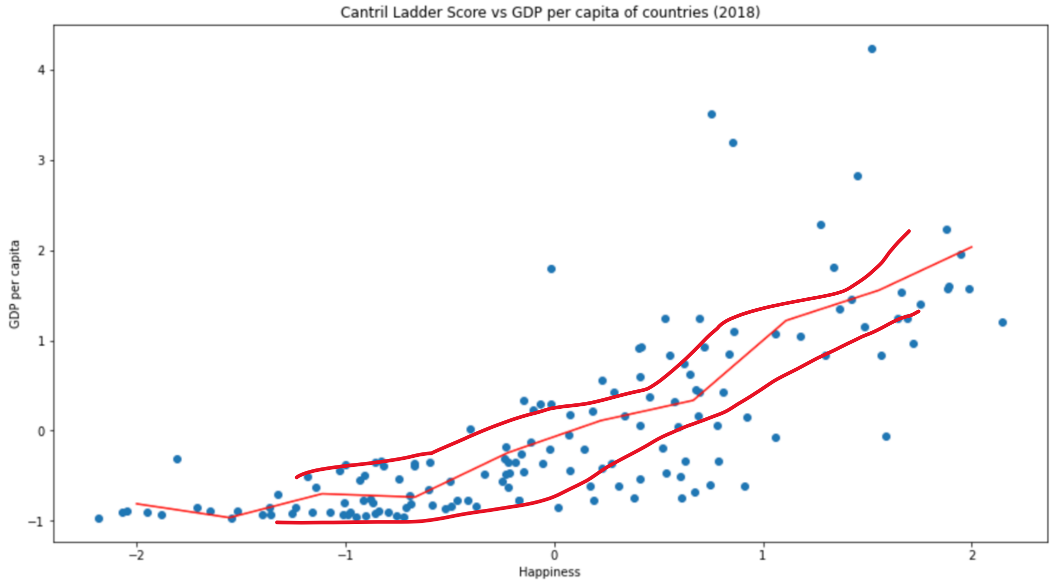
- How do we evaluate when only training data is available?
  - Using only Training MSE
- How do we evaluate when both training and test data is available?
  - Using Test MSE




```
# preprocess the inputs
X,Y = lr_ols.preprocess()

#compute beta
beta = lr_ols.train(X,Y)

# use the computed beta for prediction
Y_predict = lr_ols.predict(X,beta)

# below code displays the predicted values

# access the 1st column (the 0th column is all 1's)
X_ = X[...,1].ravel()
```

Evaluation

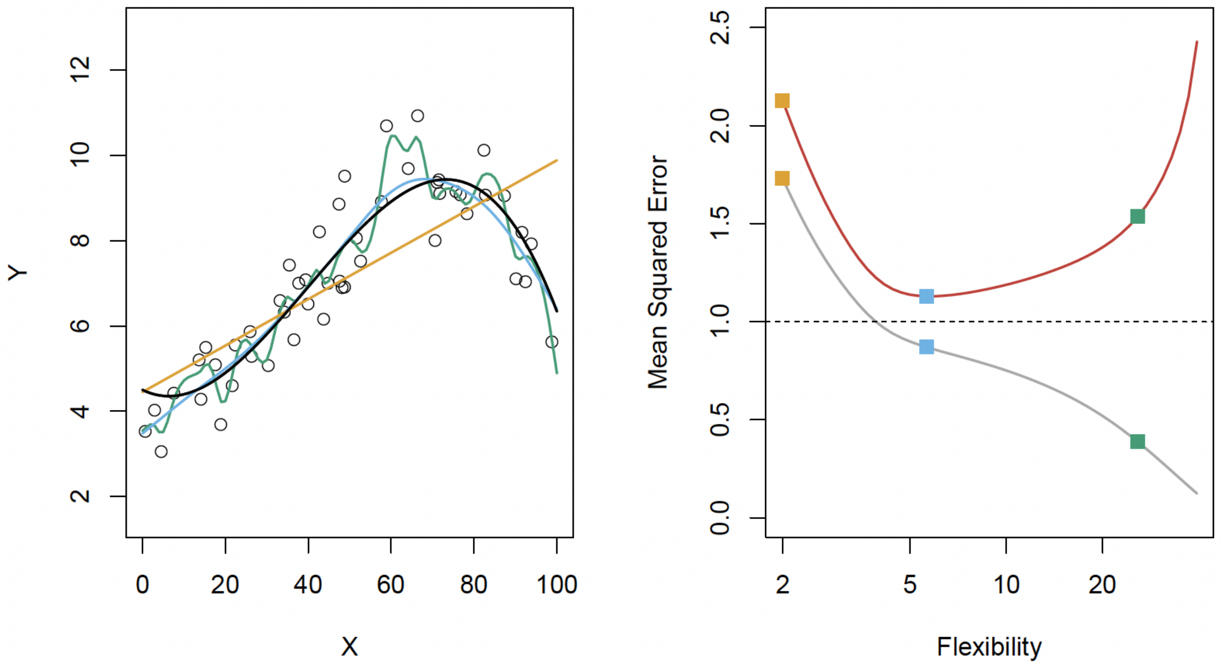
- How do we evaluate when only training data is available?
    - Using only Training MSE
- How do we evaluate when both training and test data is available?
    - Using Test MSE




```
# generate 10 random set of points between -2 and 2
x_sample = np.linspace(-2, 2, 10)

# use the computed beta for prediction
Y_test = poly_reg.predict(x_sample,beta)

# access the 0st column
X = X_[...,0].ravel()
```

Evaluation

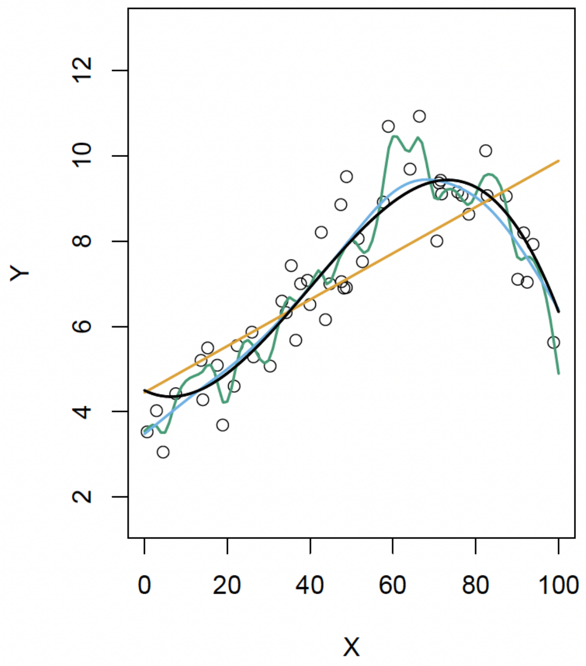
- Let’s see how train and test MSE can be different

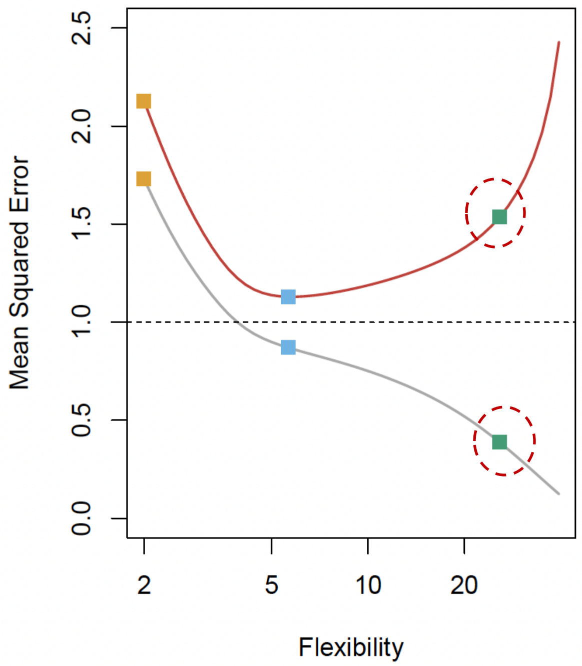


Flexibility increases in polynomial functions when n increases.

Evaluation

- **Overfitting** – When small training MSE yields large test MSE

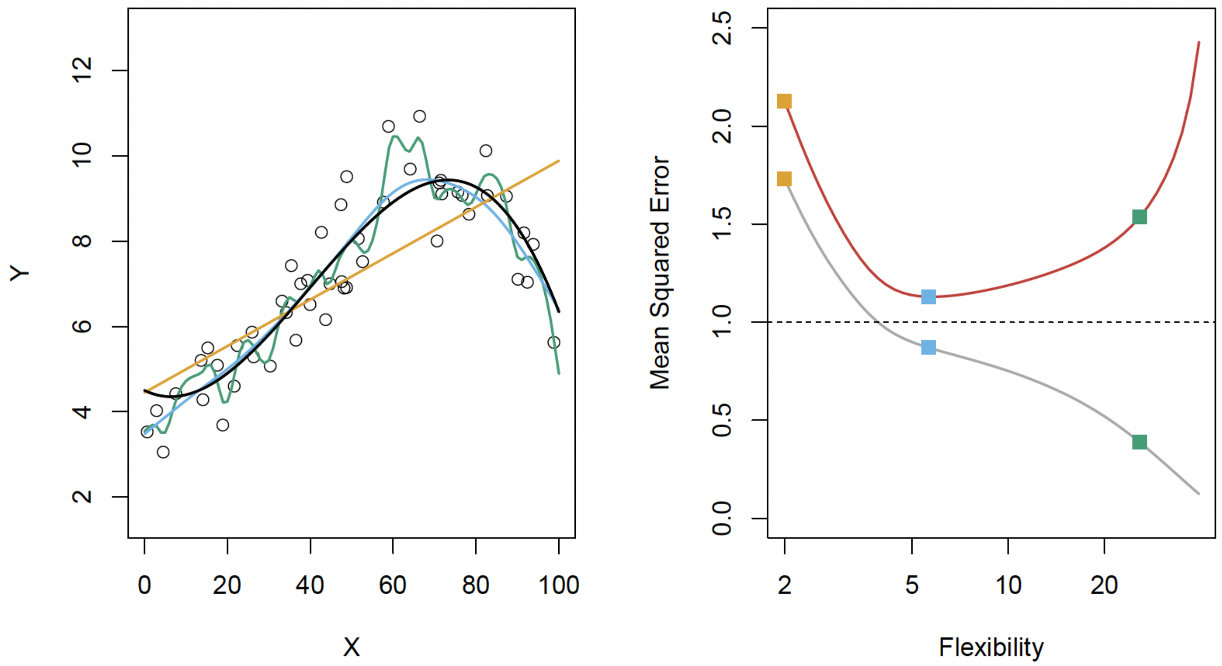




McMaster University

Evaluation

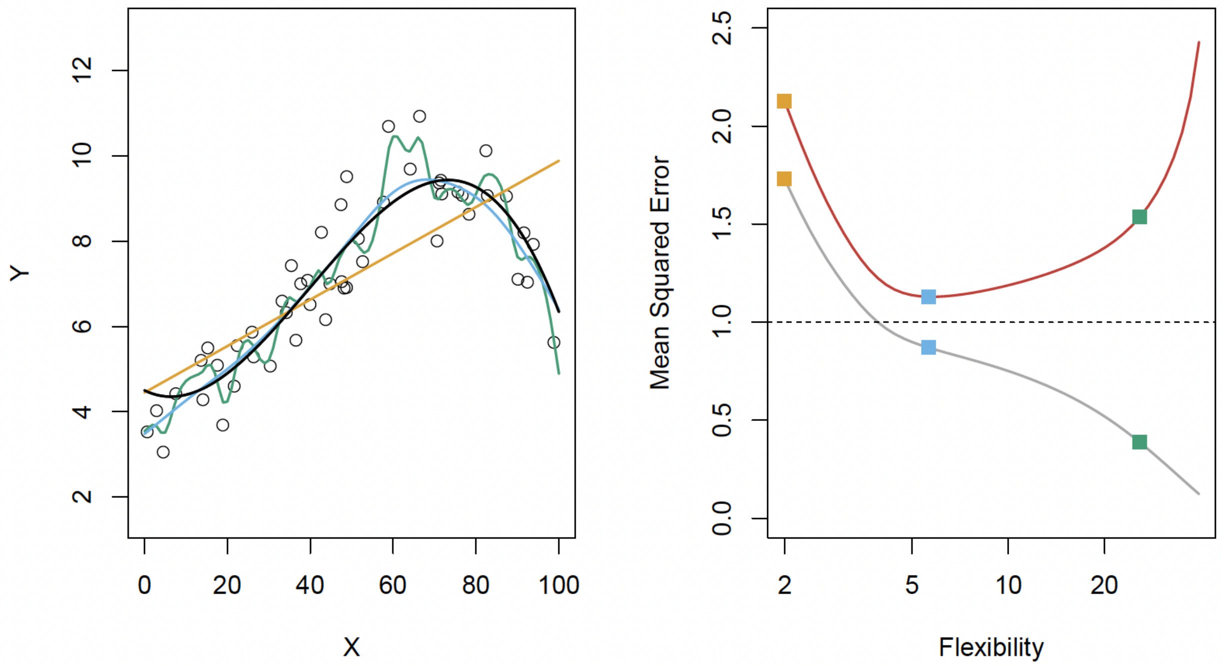
- **Overfitting** – When small training MSE yields large test MSE



Remember: We always expect the training MSE to be smaller than the test MSE, overfitting is when less flexible model would have yielded a smaller test MSE

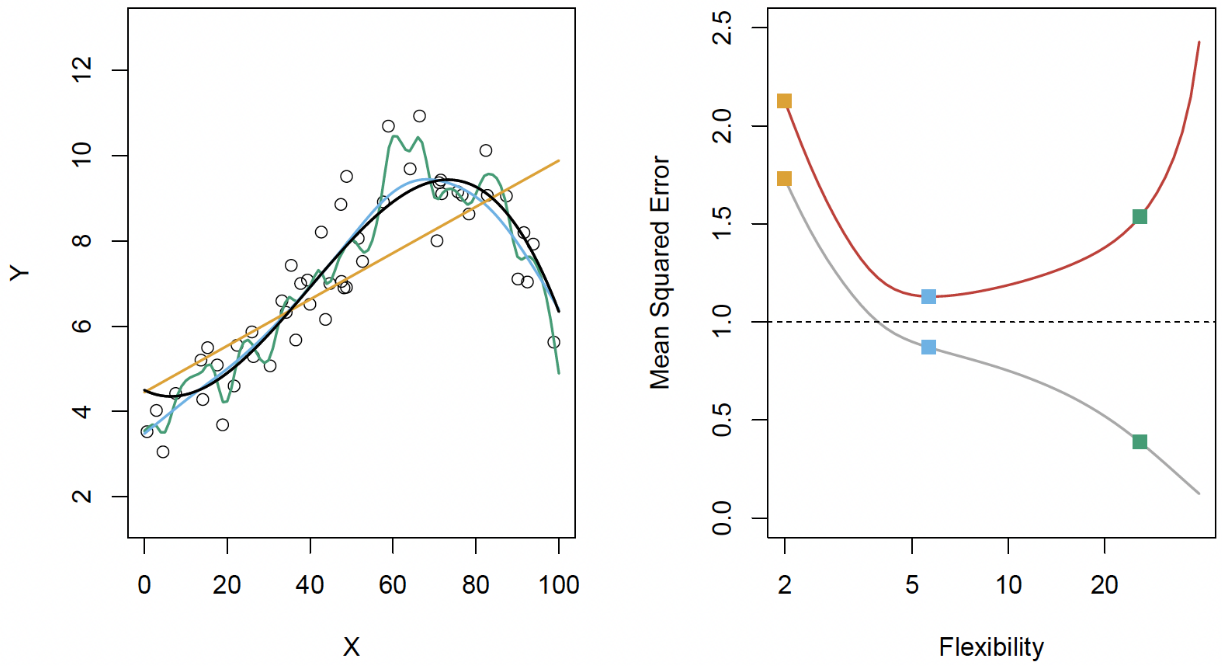
Evaluation

- **Underfitting** – When small training MSE yields small test MSE



Evaluation

- **Underfitting** – When small training MSE yields small test MSE



Remember: Underfitting occurs when a model is too simple, which can be a result of a model needing more training time, more input features, or less regularization.

McMaster University

Bias – Variance Trade Off

$$y' = \beta_0 + \beta_1 * x_1 + \beta_2 * x_1{}^2 + \beta_3 * x_1{}^3 + \cdots + \beta_n * x_1{}^n + \epsilon$$

Error term

For a given observation x_0 ,

Expected Test MSE = Variance of y' + Bias of y' + Variance of ϵ

Variance refers to the amount by which y' would change if we trained it using a different training data sets.

Bias – Variance Trade Off

$$y' = \beta_0 + \beta_1 * \mathrm{x}_1 + \beta_2 * \mathrm{x}_1{}^2 + \beta_3 * \mathrm{x}_1{}^3 + \cdots + \beta_n * \mathrm{x}_1{}^n + \epsilon$$

Error term

For a given observation x_0 ,

Expected Test MSE = Variance of y' + Bias of y' + Variance of ϵ

Bias refers to the amount by error in y' when applied to real – world problem. More flexible methods lead to less bias.

Bias – Variance Trade Off

$$y' = \beta_0 + \beta_1 * x_1 + \beta_2 * {x_1}^2 + \beta_3 * {x_1}^3 + \cdots + \beta_n * {x_1}^n + \epsilon$$

Error term

For a given observation x_0 ,

Expected Test MSE = Variance of y' + Bias of y' + Variance of ϵ

Irreducible Error

Bias – Variance Trade Off

$$y' = \beta_0 + \beta_1 * \mathrm{x}_1 + \beta_2 * \mathrm{x}_1{}^2 + \beta_3 * \mathrm{x}_1{}^3 + \cdots + \beta_n * \mathrm{x}_1{}^n + \epsilon$$

For a given observation x_0 ,

Expected Test MSE = Variance of y' + Bias of y' + Variance of ϵ

Bias – Variance Trade Off: Good test set performance requires low variance as well as low squared bias

Cross Validation

- Cross-validation can be used to estimate the test error associated with a learning method to evaluate its performance, or to select the appropriate level of flexibility.
- A good model is the one whose test error rate is the smallest.
- Cross validation approaches: Validation Set Approach
    - Divide the data into **training** set and **validation** set. Train on the training set, compute performance on the validation set.

McMaster University

Cross Validation

- Cross-validation can be used to estimate the test error associated with a learning method to evaluate its performance, or to select the appropriate level of flexibility.
- A good model is the one whose test error rate is the smallest.
- Cross Validation is done in 2 ways: Validation Set Approach
    - Divide the data into **training** set and **validation** set. Train on the training set, compute performance on the validation set.
    - Limitations of this approach:
        - High variability in the test MSE performance, and it depends, which observations are selected in the training set.
        - Number of training set observations significantly reduces.

Cross Validation

- Cross-validation can be used to estimate the test error associated with a learning method to evaluate its performance, or to select the appropriate level of flexibility.
- A good model is the one whose test error rate is the smallest.
- Cross validation approaches : Leave One Out Cross-Validation
  - Divide the data into **training** set and 1 **validation** observation. Train on the training set, compute performance on the validation set. Repeat with every observation as validation.

$$CV_n = \frac{1}{n}\sum_{i=1}^{n} MSE_i$$

Cross Validation

- Cross-validation can be used to estimate the test error associated with a learning method to evaluate its performance, or to select the appropriate level of flexibility.

- A good model is the one whose test error rate is the smallest.

- Cross validation approaches : Leave One Out Cross-Validation
  - Divide the data into training set and 1 validation observation. Train on the training set, compute performance on the validation set. Repeat with every observation as validation.
  - Advantages:
    - Less Bias
    - No randomness in split

Cross Validation

- Cross-validation can be used to estimate the test error associated with a learning method to evaluate its performance, or to select the appropriate level of flexibility.

- A good model is the one whose test error rate is the smallest.

- Cross validation approaches : Leave One Out Cross-Validation
  - Divide the data into training set and 1 validation observation. Train on the training set, compute performance on the validation set. Repeat with every observation as validation.
  - Advantages:
    - Less Bias
    - No randomness in split
  - Disadvantage:
    - Computation Heavy

$$CV_n = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - y'_i}{1 - h_i}\right)^2$$

where, $$h_i = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_{i'=1}^{n}(x_{i'} - \bar{x})^2}.$$

McMaster University

Cross Validation

- Cross-validation can be used to estimate the test error associated with a learning method to evaluate its performance, or to select the appropriate level of flexibility.

- A good model is the one whose test error rate is the smallest.

- Cross validation approaches : k-Fold Cross-Validation
  - Randomly dividing the set of observations into k-fold CV k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the remining $k-1$ are used for training.

k=2 (2-fold validation)

$$CV_k = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

Readings

***Required Readings*:**

Introduction to Statistical Learning

1. Chapter 5 – Section 5.1 Page 202 - 209
2. Chapter 7 – Section 7.1 Page 290-292

Deep Learning

1. Chapter 5 – Section 5.4 Page 122-130

Supplemental Readings (Not required but recommended):

Introduction to Statistical Learning

1. Chapter 2 – Section 2.2 Page 27-34

Thank You

McMaster University