Introduction to Data Science
CS61
June 12 - July 12, 2018

Dr. Ash Pahwa

Lesson 8: kNN Model Assessment

Lesson 8.1: kNN Confusion Matrix

Outline

- Assessment Methods
  - Assessment step in CRISP/DM Modeling Methodology
  - Confusion Matrix
  - Building Confusion Matrix in R
  - Building Confusion Matrix in Python

Goals of Predictive Analytics Application: Estimation or Classification

- Estimation – Regression modeling technique
  - Output is a number
    - House price
    - Product sales for next quarter
  - Criteria for assessment
    - R^2
    - Root Mean Square Error
- Classification – kNN, Naïve Bayes, Decision Trees etc. modeling techniques
  - Output is a categorical variable
    - Sports team will win or lose
    - Email is junk or not
  - Criteria for assessment
    - Confusion matrix
    - ROC Curves

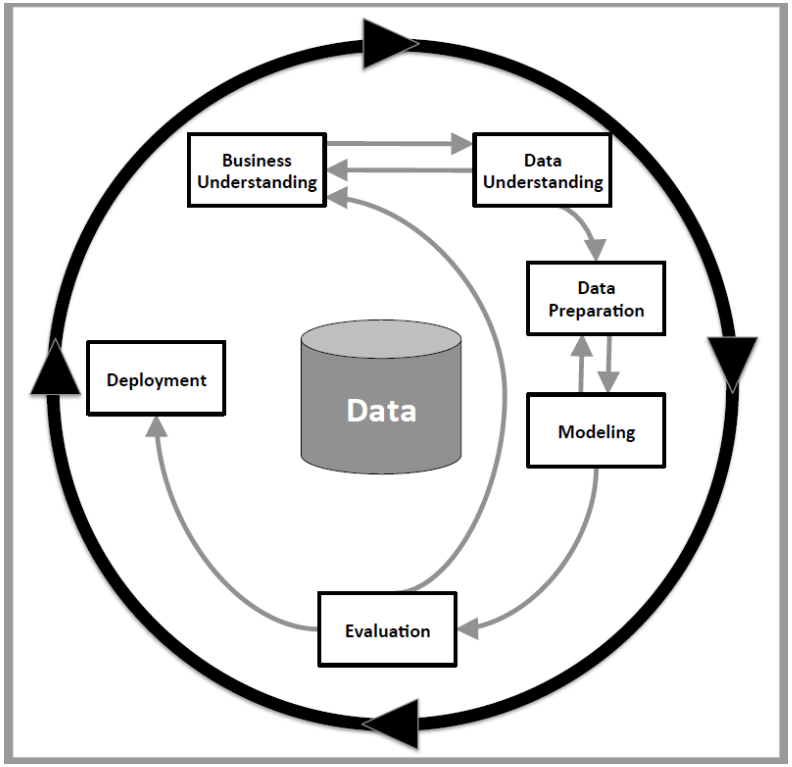
Assessment Methods

CRISP-DM Modeling Technique

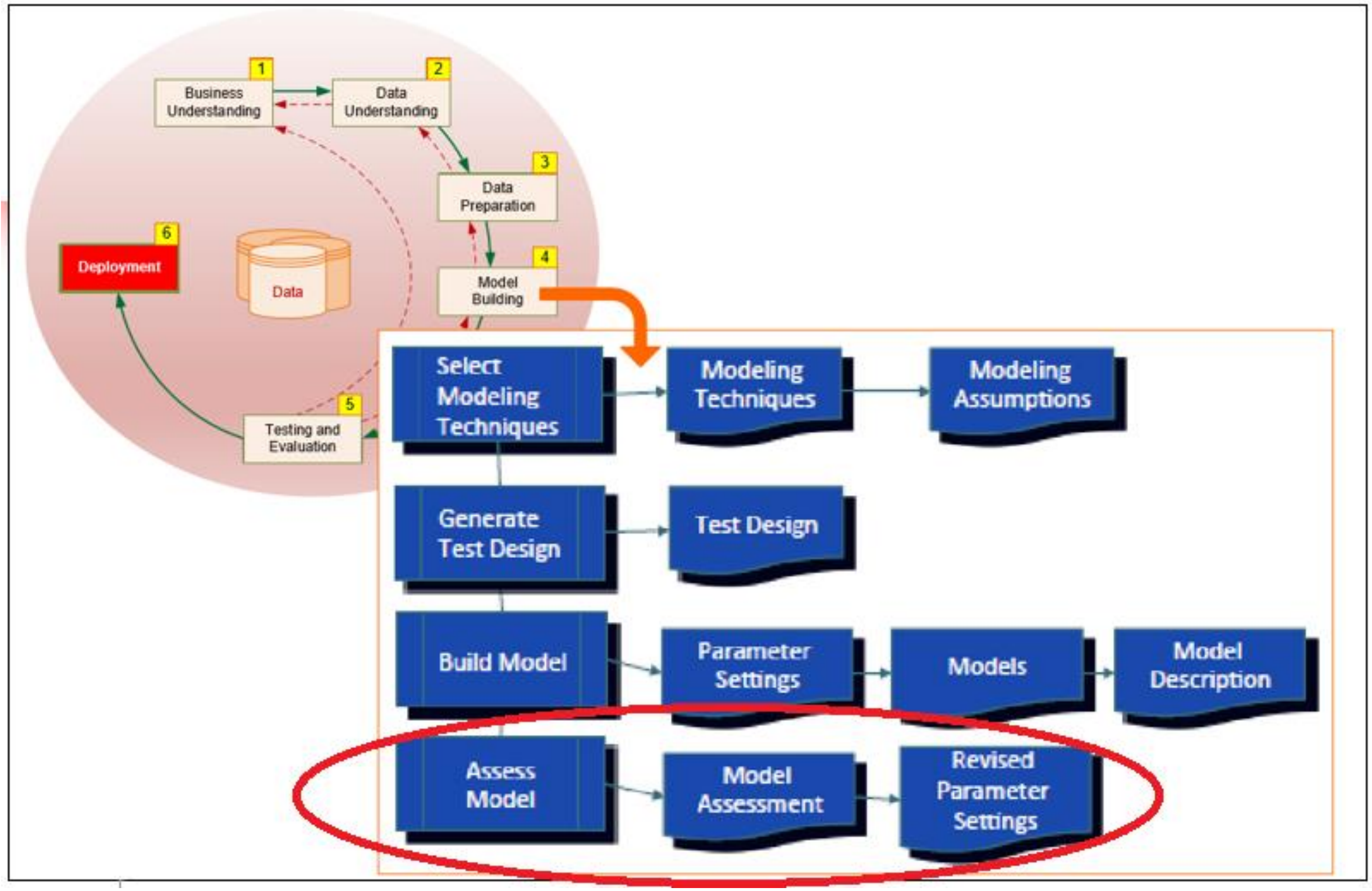
CRISP-DM Process for Modeling

- www.crisp-dm.org
- CRoss Industry Standard Process for Data Mining

The word Data Mining can be interchanged with Predictive Analytics.



Modeling – CRISP-DM Step 4



Metrics for Model Assessment

- Accuracy
  - How good does the model predict on hold-out data
- Reliability
  - Does the model do a good job on variety of data types
- Explain-ability (transparency)
  - How easy/difficult it is to explain model's logic
- Simplicity (parsimony – simpler is better)
- Theoretical foundation (optimal versus heuristic)
- Ease of deployment (and ease of maintenance)

Confusion Matrix

Confusion Matrix (Error Matrix)

- Confusion matrix: a cross tabulation of actual versus predicted number (or percent) of samples
- The name stems from the fact that it makes it easy to see if the system is confusing two classes (mislabeling one as another)

- Suppose a classification system has been trained to distinguish between cats, dogs and rabbits
- Suppose there are 27 animals
  - 8 cats
  - 6 dogs
  - 13 rabbits
- Confusion matrix may look like this
  - Out of 8 cats, 5 of them were identified correctly
  - Out of 6 dogs, 3 of them identified correctly
  - Out of 13 rabbits, 11 of them were identified correctly
  - Accuracy = (5+3+11)/27 = 19/27

		Predicted		
		Cat	Dog	Rabbit
Actual class	Cat	5	3	0
	Dog	2	3	1
	Rabbit	0	2	11

Confusion Matrix for Binary Classification

- Confusion Matrix: actual and predicted class assignment to columns and rows is arbitrary

		Predicted Class		
		Predicted as Positive	Predicted as Negative	
Actual Class	Actual Positive	TP (PP) (True Positive)	FN (PN) (False Negative)	Total Positive Actuals
	Actual Negative	FP (NP) (False Positive)	TN (NN) (True Negative)	Total Negative Actuals
		Total Positive Predictions	Total Negative Predictions	

		Predicted		
		Cat	Dog	Rabbit
Actual class	Cat	5	3	0
	Dog	2	3	1
	Rabbit	0	2	11

Binary Classification Assessment Metrics

		Predicted Class		
		Predicted as Positive	Predicted as Negative	
Actual Class	Actual Positive	TP (PP) (True Positive)	FN (PN) (False Negative)	Total Positive Actuals
	Actual Negative	FP (NP) (False Positive)	TN (NN) (True Negative)	Total Negative Actuals
		Total Positive Predictions	Total Negative Predictions	

- Using the numbers in confusion matrix, the following assessment metrics can be calculated
- $Percentage\ Correct\ Classification\ (PCC) = Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- $Accuracy = \frac{Sum\ of\ the\ diagnols}{Sum\ of\ all\ numbers\ in\ the\ confusion\ matrix}$
- $Sensitivity\ (True\ Positive)\ \&\ False\ Positive$
- $Specificity\ (True\ Negative)\ \&\ False\ Negative$

Confusion Matrix for Multi-Class Classification

- $Percent\ Correct\ Classification\ (PCC) = \frac{\sum Green\ Cells}{\sum All\ Cells}$

		Predicted Class				
		Predicted Class A	Predicted Class B	Predicted Class C	Predicted Class D	
Actual Class	Actual Class A	True Prediction A	False Prediction	False Prediction	False Prediction	Total Actual Class A
Actual Class	Actual Class B	False Prediction	True Prediction B	False Prediction	False Prediction	Total Actual Class B
Actual Class	Actual Class C	False Prediction	False Prediction	True Prediction C	False Prediction	Total Actual Class C
Actual Class	Actual Class D	False Prediction	False Prediction	False Prediction	True Prediction D	Total Actual Class D
		Total Predicted Class A	Total Predicted Class B	Total Predicted Class C	Total Predicted Class D	

		Predicted		
		Cat	**Dog**	**Rabbit**
Actual class	**Cat**	5	3	0
	Dog	2	3	1
	Rabbit	0	2	11

Building Confusion Matrix

R

Iris Dataset

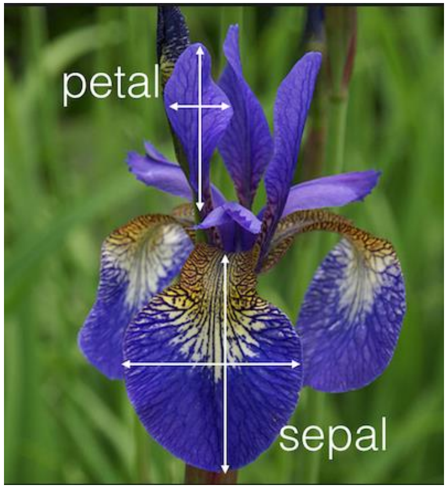
- Edgar Anderson's Iris Data
- Iris Species
  - Setosa, Versicolor, Virginica
- Sepal + Petal length and width in centimeters
- 150 records (50 flowers from each 3 species)

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

Iris Virginica

Iris Versicolor

Iris Setosa



Iris Dataset

```
> dim(iris)
[1] 150   5
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species  
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50  
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50  
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50  
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199                  
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800                  
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500                  
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> names(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
> str(iris)
'data.frame':       150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 
...
> table(iris$Species)

    setosa versicolor  virginica 
        50         50         50 
>
```

Shuffle the Dataset

```
> ####################################
> # 1. Mix all the rows
> # Like shuffle deck of cards
> #
> set.seed(9850)
> gp = runif(nrow(iris)) # Generate 150 random numbers uniformaly distributed
> iris = iris[order(gp),]
> head(iris,10)
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
103          7.1         3.0          5.9         2.1  virginica
20           5.1         3.8          1.5         0.3     setosa
63           6.0         2.2          4.0         1.0 versicolor
17           5.4         3.9          1.3         0.4     setosa
83           5.8         2.7          3.9         1.2 versicolor
53           6.9         3.1          4.9         1.5 versicolor
118          7.7         3.8          6.7         2.2  virginica
91           5.5         2.6          4.4         1.2 versicolor
80           5.7         2.6          3.5         1.0 versicolor
43           4.4         3.2          1.3         0.2     setosa
>
```

Normalize the Dataset (from 0 – 1)

```
> ######################################
> # 2. Normalize all numbers from 0 - 1
> #
> summary(iris[,c(1,2,3,4)])
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
 Median :5.800   Median :3.000   Median :4.350   Median :1.300  
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
>
> normalize = function(x) {
+   return(  (x-min(x))/(max(x)-min(x)))}
>
> iris_n = as.data.frame(lapply(iris[,c(1,2,3,4)],normalize))
> summary(iris_n)
  Sepal.Length     Sepal.Width      Petal.Length     Petal.Width     
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000  
 1st Qu.:0.2222   1st Qu.:0.3333   1st Qu.:0.1017   1st Qu.:0.08333  
 Median :0.4167   Median :0.4167   Median :0.5678   Median :0.50000  
 Mean   :0.4287   Mean   :0.4406   Mean   :0.4675   Mean   :0.45806  
 3rd Qu.:0.5833   3rd Qu.:0.5417   3rd Qu.:0.6949   3rd Qu.:0.70833  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000  
>
```

Build Training and Test Dataset

```
> #######################################
> # 3. Build Training and Test dataset
> #
> iris_train = iris_n[1:129,]
> iris_test = iris_n[130:150,]
> iris_training_target = iris[1:129,5]
> iris_test_target = iris[130:150,5]
>
```

Shuffled Test Data: Count=21

	A	B	C	D	E	F	G
1		#	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
2	1	60	5.2	2.7	3.9	1.4	versicolor
3	2	6	5.4	3.9	1.7	0.4	setosa
4	3	78	6.7	3	5	1.7	versicolor
5	4	101	6.3	3.3	6	2.5	virginica
6	5	138	6.4	3.1	5.5	1.8	virginica
7	6	21	5.4	3.4	1.7	0.2	setosa
8	7	105	6.5	3	5.8	2.2	virginica
9	8	87	6.7	3.1	4.7	1.5	versicolor
10	9	114	5.7	2.5	5	2	virginica
11	10	18	5.1	3.5	1.4	0.3	setosa
12	11	35	4.9	3.1	1.5	0.2	setosa
13	12	84	6	2.7	5.1	1.6	versicolor
14	13	29	5.2	3.4	1.4	0.2	setosa
15	14	124	6.3	2.7	4.9	1.8	virginica
16	15	106	7.6	3	6.6	2.1	virginica
17	16	115	5.8	2.8	5.1	2.4	virginica
18	17	23	4.6	3.6	1	0.2	setosa
19	18	143	5.8	2.7	5.1	1.9	virginica
20	19	129	6.4	2.8	5.6	2.1	virginica
21	20	57	6.3	3.3	4.7	1.6	versicolor
22	21	2	4.9	3	1.4	0.2	setosa
23							

Normalized Test Data: Count = 21

	A	B	C	D	E	F	G
1		**#**	**Sepal.Length**	**Sepal.Width**	**Petal.Length**	**Petal.Width**	**Species**
2	1	60	5.2	2.7	3.9	1.4	versicolor

23							
24		Minimum	4.3	2	1	0.1	
25		Maximum	7.9	4.4	6.9	2.5	
26							
27		#	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
28	1	60	0.250	0.292	0.492	0.542	versicolor
29	2	6	0.306	0.792	0.119	0.125	setosa
30	3	78	0.667	0.417	0.678	0.667	versicolor
31	4	101	0.556	0.542	0.847	1.000	virginica
32	5	138	0.583	0.458	0.763	0.708	virginica
33	6	21	0.306	0.583	0.119	0.042	setosa
34	7	105	0.611	0.417	0.814	0.875	virginica
35	8	87	0.667	0.458	0.627	0.583	versicolor
36	9	114	0.389	0.208	0.678	0.792	virginica
37	10	18	0.222	0.625	0.068	0.083	setosa
38	11	35	0.167	0.458	0.085	0.042	setosa
39	12	84	0.472	0.292	0.695	0.625	versicolor
40	13	29	0.250	0.583	0.068	0.042	setosa
41	14	124	0.556	0.292	0.661	0.708	virginica
42	15	106	0.917	0.417	0.949	0.833	virginica
43	16	115	0.417	0.333	0.695	0.958	virginica
44	17	23	0.083	0.667	0.000	0.042	setosa
45	18	143	0.417	0.292	0.695	0.750	virginica
46	19	129	0.583	0.333	0.780	0.833	virginica
47	20	57	0.556	0.542	0.627	0.625	versicolor
48	21	2	0.167	0.417	0.068	0.042	setosa
49							

Normalized Sepal Length
For observation#60

$$= \frac{5.2-4.3}{7.9-4.3} = \frac{0.9}{3.6} = 0.250$$

Normalized Sepal Width
For observation#60

$$= \frac{2.7-2.0}{4.4-2.0} = \frac{0.7}{2.4} = 0.292$$

Normalized Petal Length
For observation#60

$$= \frac{3.9-1.0}{6.9-1.0} = \frac{2.9}{5.9} = 0.492$$

Normalized Petal Width
For observation#60

$$= \frac{1.4-0.1}{2.5-0.1} = \frac{1.3}{2.4} = 0.542$$

Train the Model using Training Dataset, Predict Response Variable of the Test Dataset Package "class" has kNN module

```
> #########################################
> # call the kNN function
> #
> # The value of 'k' should be sqrt of observation
> # The value of 'k' should be be an odd number
> # If a voting tie occurs, it can be resolved
> #
> # Observations = 150
> # k = 13
> #
> library(class)
>
> k = 13
> m1 <- knn(train=iris_train, test=iris_test, cl=iris_training_target, k=k)
> (t = table(iris_test_target,m1))
                m1
iris_test_target setosa versicolor virginica
      setosa          7          0         0
      versicolor      0          3         2
      virginica       0          0         9
>
> ###################################################
> # Compute Accuracy of Prediction
> (accuracy = sum(diag(t))/sum(t)*100)
[1] 90.47619
>
```

Correct Prediction = 7 out of 7
True Class: Setosa
Predicted Class: Setosa

	A	B	C	D	E	F
1	**Sepal.Length**	**Sepal.Width**	**Petal.Length**	**Petal.Width**	**target**	**predicted**
2	0.250	0.292	0.492	0.542	versicolor	versicolor
3	0.306	0.792	0.119	0.125	setosa	setosa
4	0.667	0.417	0.678	0.667	versicolor	virginica
5	0.556	0.542	0.847	1.000	virginica	virginica
6	0.583	0.458	0.763	0.708	virginica	virginica
7	0.306	0.583	0.119	0.042	setosa	setosa
8	0.611	0.417	0.814	0.875	virginica	virginica
9	0.667	0.458	0.627	0.583	versicolor	versicolor
10	0.389	0.208	0.678	0.792	virginica	virginica
11	0.222	0.625	0.068	0.083	setosa	setosa
12	0.167	0.458	0.085	0.042	setosa	setosa
13	0.472	0.292	0.695	0.625	versicolor	virginica
14	0.250	0.583	0.068	0.042	setosa	setosa
15	0.556	0.292	0.661	0.708	virginica	virginica
16	0.917	0.417	0.949	0.833	virginica	virginica
17	0.417	0.333	0.695	0.958	virginica	virginica
18	0.083	0.667	0.000	0.042	setosa	setosa
19	0.417	0.292	0.695	0.750	virginica	virginica
20	0.583	0.333	0.780	0.833	virginica	virginica
21	0.556	0.542	0.627	0.625	versicolor	versicolor
22	0.167	0.417	0.068	0.042	setosa	setosa
23						

Correct Prediction = 3 out of 5
True Class: Versicolor
Predicted Class: Versicolor

	A	B	C	D	E	F
1	**Sepal.Length**	**Sepal.Width**	**Petal.Length**	**Petal.Width**	**target**	**predicted**
2	0.250	0.292	0.492	0.542	versicolor	versicolor
3	0.306	0.792	0.119	0.125	setosa	setosa
4	0.667	0.417	0.678	0.667	versicolor	virginica
5	0.556	0.542	0.847	1.000	virginica	virginica
6	0.583	0.458	0.763	0.708	virginica	virginica
7	0.306	0.583	0.119	0.042	setosa	setosa
8	0.611	0.417	0.814	0.875	virginica	virginica
9	0.667	0.458	0.627	0.583	versicolor	versicolor
10	0.389	0.208	0.678	0.792	virginica	virginica
11	0.222	0.625	0.068	0.083	setosa	setosa
12	0.167	0.458	0.085	0.042	setosa	setosa
13	0.472	0.292	0.695	0.625	versicolor	virginica
14	0.250	0.583	0.068	0.042	setosa	setosa
15	0.556	0.292	0.661	0.708	virginica	virginica
16	0.917	0.417	0.949	0.833	virginica	virginica
17	0.417	0.333	0.695	0.958	virginica	virginica
18	0.083	0.667	0.000	0.042	setosa	setosa
19	0.417	0.292	0.695	0.750	virginica	virginica
20	0.583	0.333	0.780	0.833	virginica	virginica
21	0.556	0.542	0.627	0.625	versicolor	versicolor
22	0.167	0.417	0.068	0.042	setosa	setosa
23						

Correct Prediction = 9 out of 9
True Class: Virginica
Predicted Class: Virginica

	A	B	C	D	E	F
1	**Sepal.Length**	**Sepal.Width**	**Petal.Length**	**Petal.Width**	**target**	**predicted**
2	0.250	0.292	0.492	0.542	versicolor	versicolor
3	0.306	0.792	0.119	0.125	setosa	setosa
4	0.667	0.417	0.678	0.667	versicolor	virginica
5	0.556	0.542	0.847	1.000	virginica	virginica
6	0.583	0.458	0.763	0.708	virginica	virginica
7	0.306	0.583	0.119	0.042	setosa	setosa
8	0.611	0.417	0.814	0.875	virginica	virginica
9	0.667	0.458	0.627	0.583	versicolor	versicolor
10	0.389	0.208	0.678	0.792	virginica	virginica
11	0.222	0.625	0.068	0.083	setosa	setosa
12	0.167	0.458	0.085	0.042	setosa	setosa
13	0.472	0.292	0.695	0.625	versicolor	virginica
14	0.250	0.583	0.068	0.042	setosa	setosa
15	0.556	0.292	0.661	0.708	virginica	virginica
16	0.917	0.417	0.949	0.833	virginica	virginica
17	0.417	0.333	0.695	0.958	virginica	virginica
18	0.083	0.667	0.000	0.042	setosa	setosa
19	0.417	0.292	0.695	0.750	virginica	virginica
20	0.583	0.333	0.780	0.833	virginica	virginica
21	0.556	0.542	0.627	0.625	versicolor	versicolor
22	0.167	0.417	0.068	0.042	setosa	setosa
23						

Errors = 2
True Class: Versicolor
Predicted Class: Virginica

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	target	predicted
0.250	0.292	0.492	0.542	versicolor	versicolor
0.306	0.792	0.119	0.125	setosa	setosa
0.667	0.417	0.678	0.667	versicolor	virginica
0.556	0.542	0.847	1.000	virginica	virginica
0.583	0.458	0.763	0.708	virginica	virginica
0.306	0.583	0.119	0.042	setosa	setosa
0.611	0.417	0.814	0.875	virginica	virginica
0.667	0.458	0.627	0.583	versicolor	versicolor
0.389	0.208	0.678	0.792	virginica	virginica
0.222	0.625	0.068	0.083	setosa	setosa
0.167	0.458	0.085	0.042	setosa	setosa
0.472	0.292	0.695	0.625	versicolor	virginica
0.250	0.583	0.068	0.042	setosa	setosa
0.556	0.292	0.661	0.708	virginica	virginica
0.917	0.417	0.949	0.833	virginica	virginica
0.417	0.333	0.695	0.958	virginica	virginica
0.083	0.667	0.000	0.042	setosa	setosa
0.417	0.292	0.695	0.750	virginica	virginica
0.583	0.333	0.780	0.833	virginica	virginica
0.556	0.542	0.627	0.625	versicolor	versicolor
0.167	0.417	0.068	0.042	setosa	setosa

	A	B	C	D	E	F
1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	target	predicted
2	0.250	0.292	0.492	0.542	versicolor	versicolor
3	0.306	0.792	0.119	0.125	setosa	setosa
4	0.667	0.417	0.678	0.667	versicolor	virginica
5	0.556	0.542	0.847	1.000	virginica	virginica
6	0.583	0.458	0.763	0.708	virginica	virginica
7	0.306	0.583	0.119	0.042	setosa	setosa
8	0.611	0.417	0.814	0.875	virginica	virginica
9	0.667	0.458	0.627	0.583	versicolor	versicolor
10	0.389	0.208	0.678	0.792	virginica	virginica
11	0.222	0.625	0.068	0.083	setosa	setosa
12	0.167	0.458	0.085	0.042	setosa	setosa
13	0.472	0.292	0.695	0.625	versicolor	virginica
14	0.250	0.583	0.068	0.042	setosa	setosa
15	0.556	0.292	0.661	0.708	virginica	virginica
16	0.917	0.417	0.949	0.833	virginica	virginica
17	0.417	0.333	0.695	0.958	virginica	virginica
18	0.083	0.667	0.000	0.042	setosa	setosa
19	0.417	0.292	0.695	0.750	virginica	virginica
20	0.583	0.333	0.780	0.833	virginica	virginica
21	0.556	0.542	0.627	0.625	versicolor	versicolor
22	0.167	0.417	0.068	0.042	setosa	setosa

All 7 Setosa identified correctly

	A	B	C	D	E	F
1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	target	predicted
2	0.250	0.292	0.492	0.542	versicolor	versicolor
3	0.306	0.792	0.119	0.125	setosa	setosa
4	0.667	0.417	0.678	0.667	versicolor	virginica
5	0.556	0.542	0.847	1.000	virginica	virginica
6	0.583	0.458	0.763	0.708	virginica	virginica
7	0.306	0.583	0.119	0.042	setosa	setosa
8	0.611	0.417	0.814	0.875	virginica	virginica
9	0.667	0.458	0.627	0.583	versicolor	versicolor
10	0.389	0.208	0.678	0.792	virginica	virginica
11	0.222	0.625	0.068	0.083	setosa	setosa
12	0.167	0.458	0.085	0.042	setosa	setosa
13	0.472	0.292	0.695	0.625	versicolor	virginica
14	0.250	0.583	0.068	0.042	setosa	setosa
15	0.556	0.292	0.661	0.708	virginica	virginica
16	0.917	0.417	0.949	0.833	virginica	virginica
17	0.417	0.333	0.695	0.958	virginica	virginica
18	0.083	0.667	0.000	0.042	setosa	setosa
19	0.417	0.292	0.695	0.750	virginica	virginica
20	0.583	0.333	0.780	0.833	virginica	virginica
21	0.556	0.542	0.627	0.625	versicolor	versicolor
22	0.167	0.417	0.068	0.042	setosa	setosa
23						

All 3 Versicolor identified correctly

	A	B	C	D	E	F
1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	target	predicted
2	0.250	0.292	0.492	0.542	versicolor	versicolor
3	0.306	0.792	0.119	0.125	setosa	setosa
4	0.667	0.417	0.678	0.667	versicolor	virginica
5	0.556	0.542	0.847	1.000	virginica	virginica
6	0.583	0.458	0.763	0.708	virginica	virginica
7	0.306	0.583	0.119	0.042	setosa	setosa
8	0.611	0.417	0.814	0.875	virginica	virginica
9	0.667	0.458	0.627	0.583	versicolor	versicolor
10	0.389	0.208	0.678	0.792	virginica	virginica
11	0.222	0.625	0.068	0.083	setosa	setosa
12	0.167	0.458	0.085	0.042	setosa	setosa
13	0.472	0.292	0.695	0.625	versicolor	virginica
14	0.250	0.583	0.068	0.042	setosa	setosa
15	0.556	0.292	0.661	0.708	virginica	virginica
16	0.917	0.417	0.949	0.833	virginica	virginica
17	0.417	0.333	0.695	0.958	virginica	virginica
18	0.083	0.667	0.000	0.042	setosa	setosa
19	0.417	0.292	0.695	0.750	virginica	virginica
20	0.583	0.333	0.780	0.833	virginica	virginica
21	0.556	0.542	0.627	0.625	versicolor	versicolor
22	0.167	0.417	0.068	0.042	setosa	setosa
23						

All 9 Virginica identified correctly

	A	B	C	D	E	F
1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	target	predicted
2	0.250	0.292	0.492	0.542	versicolor	versicolor
3	0.306	0.792	0.119	0.125	setosa	setosa
4	0.667	0.417	0.678	0.667	versicolor	virginica
5	0.556	0.542	0.847	1.000	virginica	virginica
6	0.583	0.458	0.763	0.708	virginica	virginica
7	0.306	0.583	0.119	0.042	setosa	setosa
8	0.611	0.417	0.814	0.875	virginica	virginica
9	0.667	0.458	0.627	0.583	versicolor	versicolor
10	0.389	0.208	0.678	0.792	virginica	virginica
11	0.222	0.625	0.068	0.083	setosa	setosa
12	0.167	0.458	0.085	0.042	setosa	setosa
13	0.472	0.292	0.695	0.625	versicolor	virginica
14	0.250	0.583	0.068	0.042	setosa	setosa
15	0.556	0.292	0.661	0.708	virginica	virginica
16	0.917	0.417	0.949	0.833	virginica	virginica
17	0.417	0.333	0.695	0.958	virginica	virginica
18	0.083	0.667	0.000	0.042	setosa	setosa
19	0.417	0.292	0.695	0.750	virginica	virginica
20	0.583	0.333	0.780	0.833	virginica	virginica
21	0.556	0.542	0.627	0.625	versicolor	versicolor
22	0.167	0.417	0.068	0.042	setosa	setosa
23						

Errors = 2

```
                  m1
iris_test_target setosa versicolor virginica
      setosa          7          0         0
      versicolor      0          3         2
      virginica       0          0         9
```

Building Confusion Matrix

Python

Load the Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier

from sklearn.cross_validation import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
```

Read the Iris Dataset

```
iris = load_iris()
ir = pd.DataFrame(iris.data)
ir.columns = iris.feature_names
ir['Class'] = iris.target

ir.describe()
Out[14]:
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.054000           3.758667
std             0.828066          0.433594           1.764420
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)       Class
count        150.000000  150.000000
mean           1.198667    1.000000
std            0.763161    0.819232
min            0.100000    0.000000
25%            0.300000    0.000000
50%            1.300000    1.000000
75%            1.800000    2.000000
max            2.500000    2.000000
```

Split Data into Train + Test

```
#############################################
#

x_train, x_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.14)
```

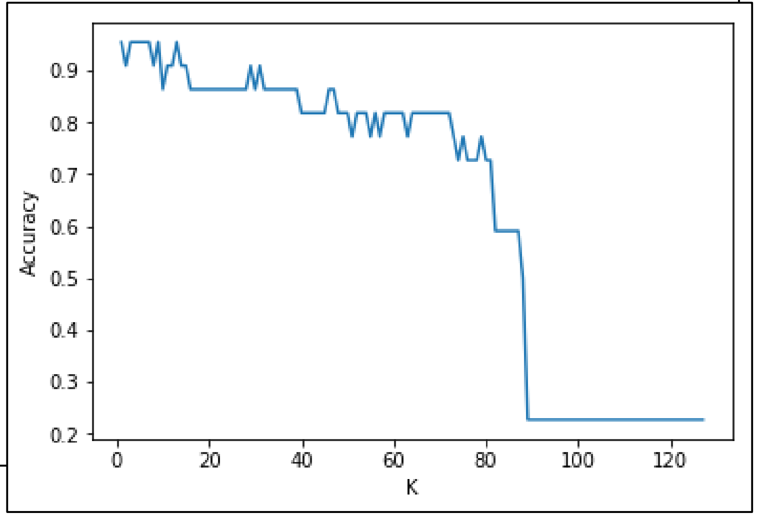
Build Model Using Training Data
Predict Using Testing Data
Compute Accuracy + Build Confusion Matrix

```
##################################################################
# Classifier kNN
#

clf = KNeighborsClassifier(n_neighbors=3).fit(x_train, y_train)

y_predict = clf.predict(x_test)

accuracy_score(y_test, y_predict)
Out[24]: 0.9545454545454545459

confusion_matrix(y_test,y_predict)
Out[25]:
array([[8, 0, 0],
       [0, 5, 0],
       [0, 1, 8]], dtype=int64)
```

- Accuracy = (8+5+8)/22=95.45%

Find Value of 'k' when Accuracy is Maximum

```
accuracy_values=[]
k_values=[]

x_train.shape[0]
Out[28]: 128

for x in range(1,x_train.shape[0]):
    clf = KNeighborsClassifier(n_neighbors=x).fit(x_train, y_train)
    y_predict = clf.predict(x_test)
    accuracy = accuracy_score(y_test, y_predict)
    accuracy_values.append(accuracy)
    k_values.append(x)
    pass

accuracy_values = np.array(accuracy_values)

k_values = np.array(k_values)

plt.plot(k_values,accuracy_values)
plt.xlabel("K")
plt.ylabel("Accuracy")
```



Summary

- Assessment Methods
  - Assessment step in CRISP/DM Modeling Methodology
  - Confusion Matrix
  - Building Confusion Matrix in R
  - Building Confusion Matrix in Python