Regression Analysis

Regression Analysis in Practice

Nicoleta Serban, Ph.D.
Professor
School of Industrial and Systems Engineering

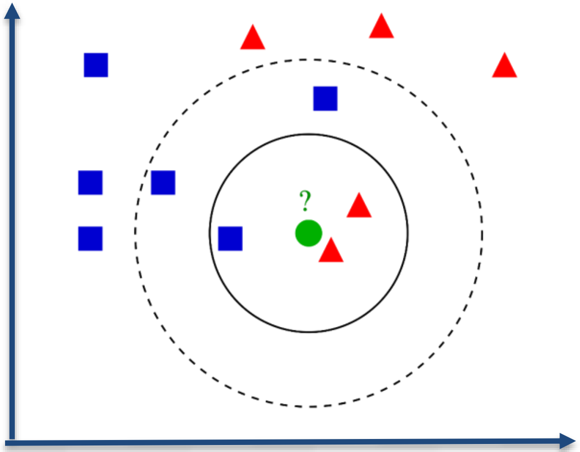
Predicting Customer Churn using Other Modeling Techniques

About This Lesson

K Nearest Neighbors (KNN): Introduction

- Classify new observations, in this case customers, according to K most similar observations.
- Class of the new observations = most found class among K nearest observations
- Supervised Learning: The labels of some observations (churn values for some customers) are known.
- Requires definition of a similarity measure (distance).

Assume that we have only two continuous features for the churn dataset. The plot represents the customer with known labels and a new customer with no information on customer’s churn value



▲ : Customers who do churn

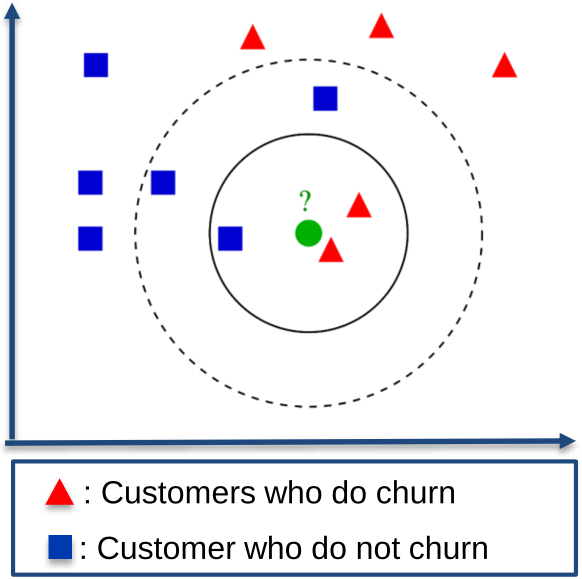
■ : Customer who do not churn

Assume the green dot represents the new customer and the contour lines represent the equal distance
from the green dot.
How do we classify the new customer if

- K = 3?
- K = 5?

K Nearest Neighbors (KNN): Introduction

- Classify new observations, in this case customers, according to K most similar observations.
- Class of the new observations = most found class among K nearest observations
- Supervised Learning: The labels of some observations (churn values for some customers) are known.
- Requires definition of a similarity measure (distance).



Defining Similarity

Assuming all features are continuous variables, let $X_{new} \in \mathbb{R}^p$ define the feature vector of new observation and X_i define the feature vectors of all available data where $i \in \{1,2,\dots,N\}$. We find the similarity between observations (customers) using

$Similarity\ between\ the\ new\ customer\ and\ i^{th}\ customer$

$$= (\sum_{k=1}^{p}(|X_{new}^{k} - X_i^{k}|)^q)^{1/q}$$

If

- q = 1 => Manhattan distance
- q = 2 => Euclidean distance
- q ∈ $\mathbb{R}^+ \cup \{0\}$ => Minkowski distance

If features are categorical,

$Similarity\ between\ the\ new\ customer\ and\ i^{th}\ customer$

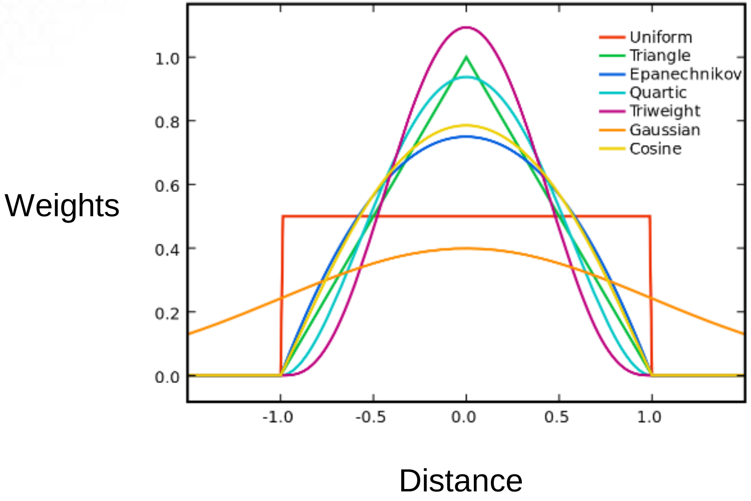
$= D_H = \sum_{i=1}^{p} D_i$ where

$$D_i = 0\ if\, X_{new}^k = X_i^k$$
$$D_i = 1\ if\, X_{new}^k \neq X_i^k$$

K Nearest Neighbors (KNN): Implementation

Kernel-Weighted Average Classification

What if we would like to assign more importance to the closer (more similar) observations in classifying the new observation?

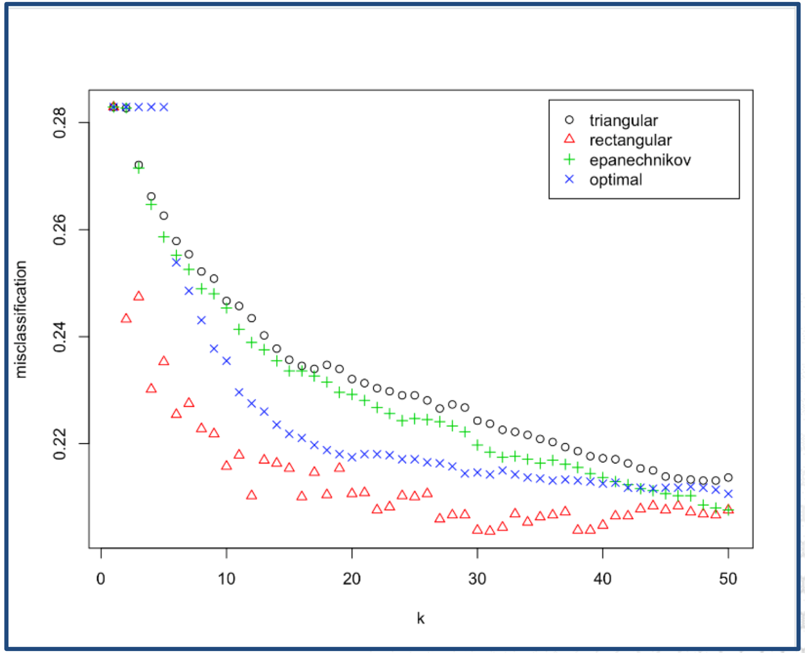


```
## Convert response to factor
y.train <- as.factor(train$`Churn Value`)
y.test <- as.factor(test$`Churn Value`)
## Dummify categorical features
dummies.train <- dummyVars(`Churn Value` ~ ., data =
train)
dummies.test <- dummyVars(`Churn Value` ~ ., data =
test)
## Create data frames containing the predictors
x.train.knn <- data.frame(predict(dummies.train, newdata =
train))
x.test.knn <- data.frame(predict(dummies.test, newdata =
test))
## Use leave-one-out cross-validation to find the
optimal value of "k"
(kknn.train <- train.kknn(y.train ~ ., x.train.knn, kmax = 50,
       kernel = c("triangular", "rectangular",
                  "epanechnikov", "optimal"),
                   scale = TRUE))
## Plot of missclassification errors vs. k for different
kernels
plot(kknn.train)
```

K Nearest Neighbors (KNN): Fitting

Leave-one-out CV

- Leave one data point out and fit a KNN model given the kernel (uniform, triangle etc.) and the value of parameter K
- Find whether the model classifies the data point correctly
- Apply the same method for all data points
- Find the misclassification rate = incorrectly classified data points / number of all data points



Optimal K = 31 and kernel = rectangular

K Nearest Neighbors (KNN): Prediction

```
# Predict the labels on the test set using Rectangular kernels and K=31
pred.knn <- predict(kknn.train, x.test.knn)

#  Calculate classification Evaluation Metrics
pred_metrics("KNN", y.test, pred.knn)
```

```
                           KNN
     Accuracy       Sensitivity       Specificity
     0.7957907      0.8606811         0.6158798
```

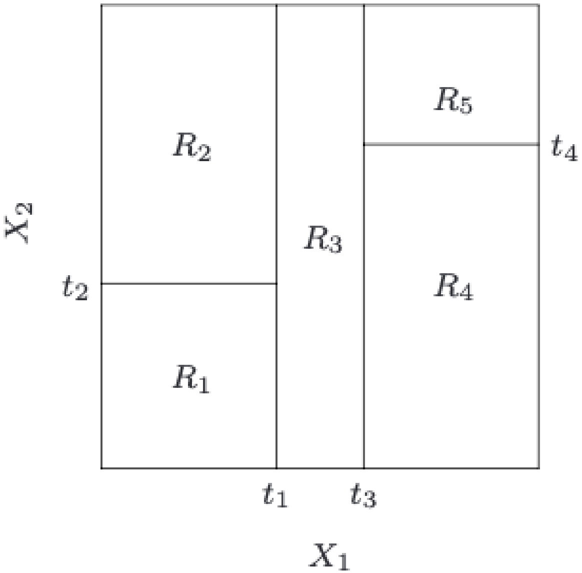
Chosen KNN model is more successful in identifying the people who churn compared to identifying people who do not churn.

Decision Trees: Introduction

Decision Trees (DT) are non-parametric supervised learning models used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

- Classification tree partitions the feature space into a set of rectangles
- Fit a simple model (like a constant) in each one

In our case of classification, partitioning occurs according to a specific rule and each rectangle takes the value 0 or 1 according to some other specific rule.



Partitioning in a feature space ∈ $\mathbb{R}^2$ with two features

Greedy Procedure for partitioning and classification under continuous features:

- Consider a splitting variable j and split point $s \in \mathbb{R}$ and define half-spaces

$$R_1(j,s) = \{X|X_J \leq s\} \,\&\, R_2(j,s) = \{X|X_J > s\}$$

- Solve the problem

$$\min_{j,s}[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2]$$

$$\hat{c}_1 = majority(y_i|x_i \in R_1(j,s))$$

$$\hat{c}_2 = majority(y_i|x_i \in R_2(j,s))$$

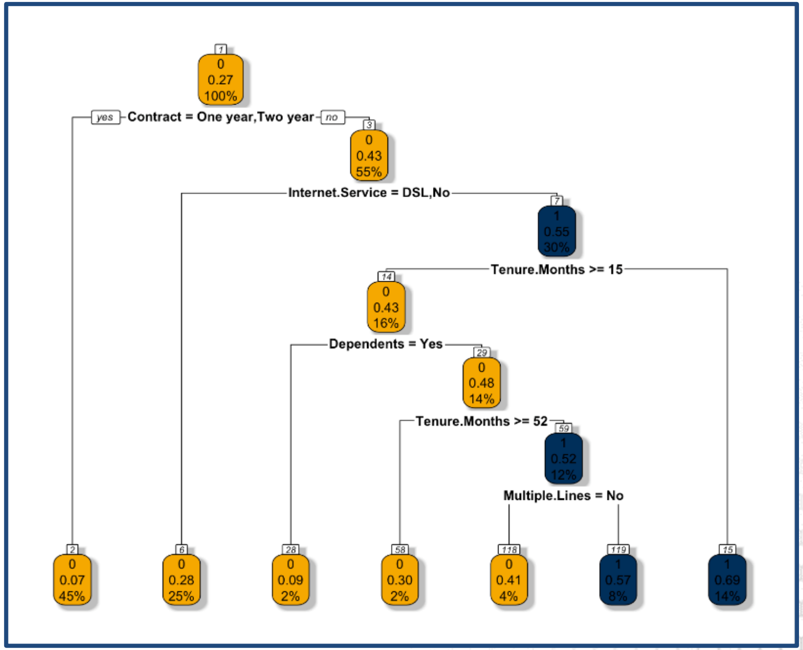
Decision Trees: Implementation

```
## Building model
set.seed(300)
decision_tree <- rpart(Churn.Value~., data = train, method = "class")

## Plotting model
rpart.plot(decision_tree, box.palette = c(buzzgold, gtblue), shadow.col = "gray", nn=TRUE)
```

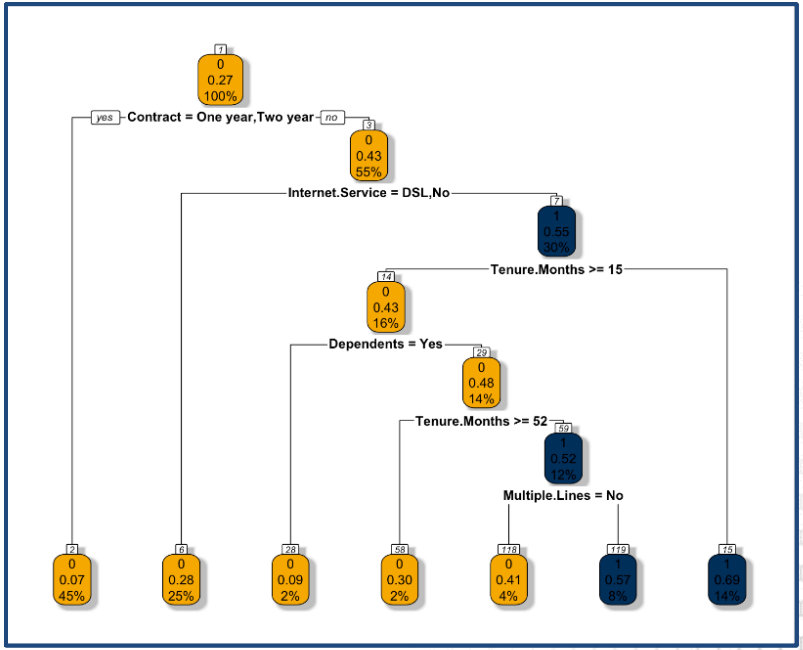
How to read the decision tree?

The first number in the node corresponds to the classification of the node (0 if not churn and 1 if churn). The second number in the node corresponds to the predicted probability of churn. The third value in the node measures the total % of customers that are included in that node.



Decision Trees: Interpretation

- The most important variable in determining churn rate is duration of contract. If the contract is 1 year, or 2 years the probability to not churn is 93%. The probability of not churning is lower if the contract is month-to-month. 45% of the total customers in the testing dataset fall in this category.
- If the customer has a month-to-month contract, has fiber optic, is in default for more than 15 months, and has dependents, then the probability of churn is only 9% ,with 2% of customers in this node.
- The higher churn occurs for month-to-month contracts, fiber optic, tenure higher than 15 months but lower than 52 months, no dependents and multiple lines. In that case, churn rate is 57%.
- Overall, the probabilities of churn are high for month-to-month contracts. The company can create incentives for customers to subscribe to longer contracts.



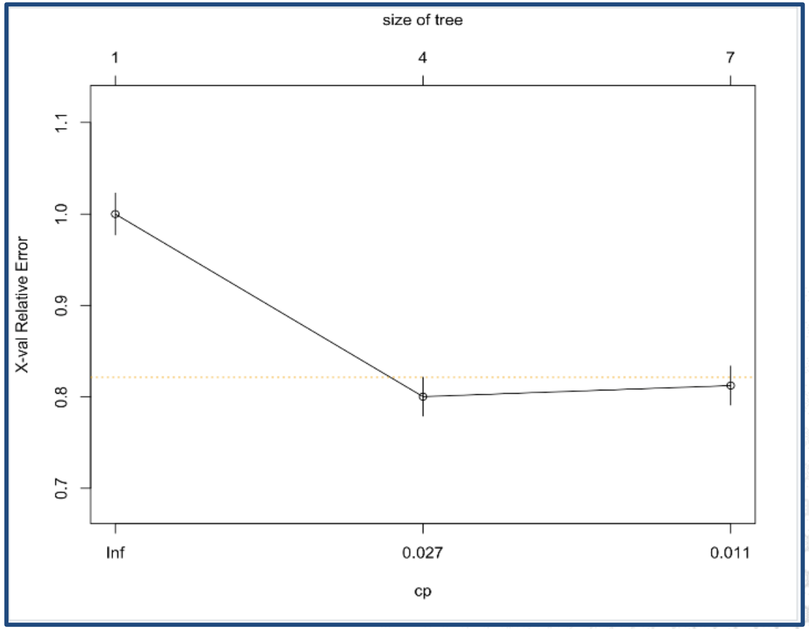
Decision Trees: Prediction

```
## Visualize cross-validation in table format
printcp(decision_tree)

## Plot the complexity parameter table
plotcp(decision_tree,minline = TRUE, lty = 3,col = buzzgold)

## Predict Churn Score
predicted_churn_score <-
predict(decision_tree,test,type="class")

## Create Confusion Matrix
confusionMatrix(data = as.factor(predicted_churn_score),
reference = as.factor(test$Churn.Value), positive = "1")
```



- Complexity Parameter finds the size of the tree that balances the size of the tree and the goodness of fit
- Note that tree size 7 minimizes the CP

Prediction\Actual	0	1
0	1220	284
1	71	183

Random Forest: Implementation

- A main issue of the tree-based method is large variance
- Trees (if grown deep enough) have low bias: can capture complicated structure but are known to be very noisy (Bias-variance tradeoff)
- Random forest: averaging

```
## Build Random Forest Model
rf <- randomForest(factor(Churn.Value)~.,data = train)

## Predict using Random Forest Model
pred_test <- predict(rf, test, type="class")

## Confusion Matrix
rf$confusion
accuracy_rf <-
(rf$confusion[1,1]+rf$confusion[2,2])/(rf$confusion[1,1]+rf$confusion[1,2]+rf$confusion[2,1]+rf$confusion[2,2])
accuracy_rf
pred_metrics("Random Forest Model",test$Churn.Value,
pred_test)
```

Random Forest		
Accuracy	Sensitivity	Specificity
0.7969283	0.8384058	0.6455026

Prediction\Actual	0	1
0	3515	357
1	684	718

In this case, the accuracy of the random forest model was just slightly better than decision tree.

Conclusion

Full Model

Accuracy	Sensitivity	Specificity
0.8139932	0.8528551	0.6785714

Stepwise Regression Model

Accuracy	Sensitivity	Specificity
0.8134243	0.8532649	0.6759494

Lasso Regression Model

Accuracy	Sensitivity	Specificity
0.8156997	0.8536942	0.6828645

Elastic Net Regression Model

Accuracy	Sensitivity	Specificity
0.8156997	0.8526623	0.6847545

Decision Tree

Accuracy	Sensitivity	Specificity
0.798066	0.8111702	0.7204724

KNN

Accuracy	Sensitivity	Specificity
0.7957907	0.8606811	0.6158798

Random Forest

Accuracy	Sensitivity	Specificity
0.7969283	0.8384058	0.6455026

From the classification metrics above, we can see that both the Lasso Regression and Elastic Regression models have slightly better metrics than the other models. Therefore, those could be the chosen ones to continue to tune and work with.

Georgia Tech

Summary