**Core Ideas in Data Mining**

1. **Classification: divide into 2 or more classes**

The recipient of an offer can respond or not respond. An applicant for a loan can repay on time, repay late, or declare bankruptcy. A credit card transaction can be normal or fraudulent.

1. **Prediction (common term** *regression)***:** book refers to the prediction of the value of a continuous variable.

trying to predict the value of a numerical variable (e.g., amount of purchase) rather than a class (e.g.,purchaser or nonpurchaser).

1. **Association Rules and Recommendation Systems**

“what goes with what.” *Association rules*, or *affinity analysis*

***collaborative filtering***, a method that uses individual users’ preferences and tastes given their historic purchase, rating, browsing, or any other measurable behavior indicative of preference, as well as other users’ history.

In contrast to ***association rules***that generate rules general to an entire population, collaborative filtering generates “what goes with what” at the individual user level.

1. **Predictive Analytics:** *clustering*

The term predictive analytics is sometimes used to also include data pattern identification methods such as clustering.

1. **Data Reduction and Dimension Reduction**

**Data Reduction:** This process of consolidating a large number of records (or cases) into a smaller set.

**Dimension Reduction:** Reducing the number of variables

The performance of data mining algorithms is often improved when the number of variables is limited, and when large numbers of records can be grouped into homogeneous groups. For example, rather than dealing with thousands of product types, an analyst might wish to group them into a smaller number of groups and build separate models for each group. Or a marketer might want to classify customers into different “personas,” and must therefore group customers into homogeneous groups to define the personas. This process of consolidating a large number of records (or cases) into a smaller set is termed ***data reduction*.** Methods for reducing the number of cases are often called *clustering*.

1. **Supervised and Unsupervised Learning**

**Supervised (labeled data)**: must have data available in which the value of the outcome of interest (e.g., purchase or no purchase) is known.

Train->validate-> test

Eg: simple linear regression ( find Y based on X- min squared deviation to obtain most accurate model output i.e. less r^2 more accurate model output)

**Unsupervised**: are those used where there is no outcome variable to predict or classify. Hence, there is no “learning” from cases where such an outcome variable is known. Association rules, dimension reduction methods, and clustering techniques are all unsupervised learning methods.

Supervised and unsupervised methods are sometimes used in conjunction. For example, unsupervised clustering methods are used to separate **loan applicants into several risk-level groups**. Then, supervised algorithms are **applied separately to each risk-level group for predicting propensity** of loan default.

**Data analysis:**

housing.df <- read.csv("WestRoxbury.csv", header = TRUE) # load data

dim(housing.df) # find the dimension of data frame

head(housing.df) # show the first six rows

View(housing.df) # show all the data in a new tab

# Practice showing different subsets of the data

# **df[row, column]**

housing.df[1:10, 1] # show the first 10 rows of the first column only

housing.df[1:10, ] # show the first 10 rows of each of the columns

housing.df[5, 1:10] # show the fifth row of the first 10 columns

housing.df[5, c(1:2, 4, 8:10)] # show the fifth row of some columns

housing.df[, 1] # show the whole first column

housing.df$TOTAL\_VALUE # a different way to show the whole first column

housing.df$TOTAL\_VALUE[1:10] # show the first 10 rows of the first column

length(housing.df$TOTAL\_VALUE) # find the length of the first column

mean(housing.df$TOTAL\_VALUE) # find the mean of the first column

summary(housing.df) # find summary statistics for each column Data

**Sampling data:**

# random sample of 5 observations

s <- sample(row.names(housing.df), 5)

housing.df[s,]

# oversample houses with over 10 rooms

s <- sample(row.names(housing.df), 5, prob = ifelse(housing.df$ROOMS>10, 0.9, 0.01))

housing.df[s,]

oversampling is required in case of rare events which happen one in 100s

**Outlier:**

**Anything away from 3 standard deviation from mean is the outlier.**

**Missing Values:**

1. **Delete empty records**
2. **Replace with mean, median**

> housing.df[rows.to.missing,]$BEDROOMS <- NA

> summary(housing.df$BEDROOMS)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

1.00 3.00 3.00 3.23 4.00 9.00 10

> housing.df[rows.to.missing,]$BEDROOMS <- median(housing.df$BEDROOMS, na.rm = TRUE)

> summary(housing.df$BEDROOMS)

Min. 1st Qu. Median Mean 3rd Qu. Max.

1.00 3.00 3.00 3.23 4.00 9.00

1. **For sensitive data (medical records, financial outcomes) use linear regression**

**Normalizing (Standardizing) & Rescaling Data:** bringing measures to one scale. Eg days-months to days

**Normalizing**: To normalize a variable, we subtract the mean from each value and then divide by the standard deviation. This operation is also sometimes called ***standardizing***. In R, function ***scale()***performs this operation. In effect, we are expressing each value as the “number of standard deviations away from the mean,” also called a ***z-score*.**

z-score= (each data record - mean)/standard deviation

**Rescaling**: rescaling each variable to a [0,1] scale. This is done by subtracting the minimum value and then dividing by the range. Subtracting the minimum shifts the variable origin to zero. Dividing by the range shrinks or expands the data to the range [0,1]. In R, rescaling can be done using function ***rescale****()* in the scales package.

**Cross Validation:**

When the dataset is really small, randomly divide it into k parts use k-1 folds as training and one as validation.

**code for partitioning the data into training, validation (and test) sets**

# use set.seed() to get the same partitions when re-running the R code.

set.seed(1)

## **partitioning into training (60%) and validation (40%)**

train.rows<-sample(rownames(housing.df), dim(df)[1]\*0.6)

train.data <- housing.df[train.rows, ]

valid.rows <-setdiff(rownames(housing.df), train.rows)

valid.data <- housing.df[valid.rows, ]

**## partitioning into training (50%), validation (30%), test (20%)**

train.rows <- sample(rownames(housing.df), dim(housing.df)[1]\*0.5)

valid.rows <- sample(setdiff(rownames(housing.df), train.rows), dim(housing.df)[1]\*0.3)

test.rows <- setdiff(rownames(housing.df), union(train.rows, valid.rows))

train.data <- housing.df[train.rows, ]

valid.data <- housing.df[valid.rows, ]

test.data <- housing.df[test.rows, ]

**fitting regression model on training data**

reg<- lm(total\_value~., data=housing.df , subset=train.rows)

tr.res<- data.frame( train.data$total\_value, reg$fitted.values, reg$residuals)

head(tr.res)

**train.data.TOTAL\_VALUE reg.fitted.values reg.residuals**

371.6 371.5818 0.018235205

299.4 299.4014 -0.001431463

294.5 294.4762 0.023835688

249.4 249.4029 -0.002874472

505.5 505.5246 -0.024612237

410.5 410.5323 -0.032339156

**code for applying the regression model to predict validation set**

pred<-predict(reg, newdata=valid.data)

vl.res<- data.frame(valid.data$total\_value, pred, residuals=valid.data$total\_value – pred)

head(vl.res)

valid.data.TOTAL\_VALUE pred residuals

344.2 344.2388 -0.038842075

575.0 575.0025 -0.002509638

298.2 298.2107 -0.010697442

313.1 313.0764 0.023567596

344.9 344.8719 0.028111825

330.7 330.7222 -0.022234883

**code for computing model evaluation metrics**

#compute accuracy for training and validation data

library(forecast)

accuracy(reg$fitted.values, train.data$total\_value)

ME RMSE MAE MPE MAPE

1.388101e-16 0.02268016 0.01956465 5.193036e-06 0.00528389

Pred<-predict(reg, newdata=valid.data)

Accuracy(pred, valid.data$total\_value)

ME RMSE MAE MPE MAPE

90.86934 161.5043 118.6455 15.14207 24.45668