STAT 408 - Statistical Learning Predictive Modeling

December 5, 2017

Statistical Learning Overview

Statistical Learning

Here are a few questions to consider:

- What does statistical learning mean to you?
- Is statistical learning different from statistics as a whole?
- What about terms like: data science, data mining, data analytics, machine learning, predictive analytics, how are these different from statistics and statistical learning?

Statistical Learning Definition

> *Statistical learning refers to a set of tools for modeling and understanding complex datasets. It is a recently developed area in statistics and blends with parallel developments in computer science and, in particular, machine learning. The field encompasses many methods such as the lasso and sparse regression, classification and regression trees, and boosting and support vector machines.*

Courtesy of *An Introduction to Statistical Learning: with Applications in R*, by Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. Note: a free e-version of this textbook can be obtain for free through the MSU Library.

Predictive Modeling

Predictive Modeling Overview

Recall the Seattle housing data set, how would you:

- Build a model to predict housing prices in King County
- Determine if your model was good or useful?

price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	zipcode
1350000	3	2.50	2753	65005	1.0	1	98070
228000	3	1.00	1190	9199	1.0	0	98148
289000	3	1.75	1260	8400	1.0	0	98148
720000	4	2.50	3450	39683	2.0	0	98010
247500	3	1.75	1960	15681	1.0	0	98032
850830	3	2.50	2070	13241	1.5	0	98102
890000	4	1.00	2550	4000	2.0	0	98109
258000	5	2.00	2260	12500	1.0	0	98032
440000	3	2.50	1910	66211	2.0	0	98024
213000	2	1.00	1000	10200	1.0	0	98024

Loss functions

A loss function is a principled way to compare a set of predictive models.

Squared Error:

$$(Price_{pred} - Price_{actual})^2$$

Zero - One Loss (binary setting):

$$f(x) = \begin{cases} 1, & \text{if } y_{pred} \neq y_{actual} \\ 0, & y_{pred} = y_{actual} \end{cases}$$

Model Evaluation

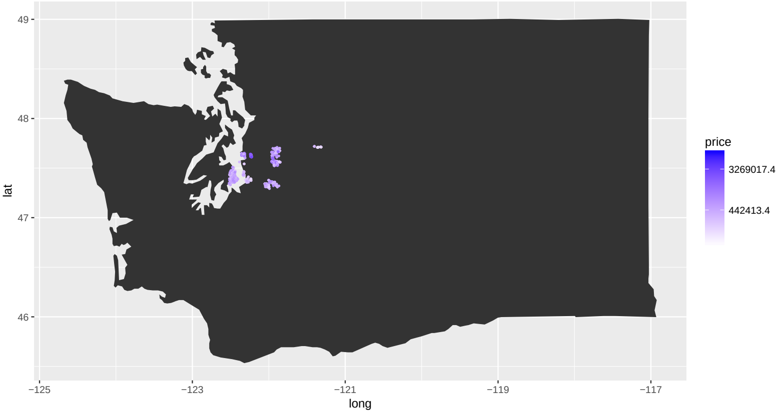
STAT 408 - Statistical Learning Predictive Modeling

Statistical Learning Overview

Predictive Modeling

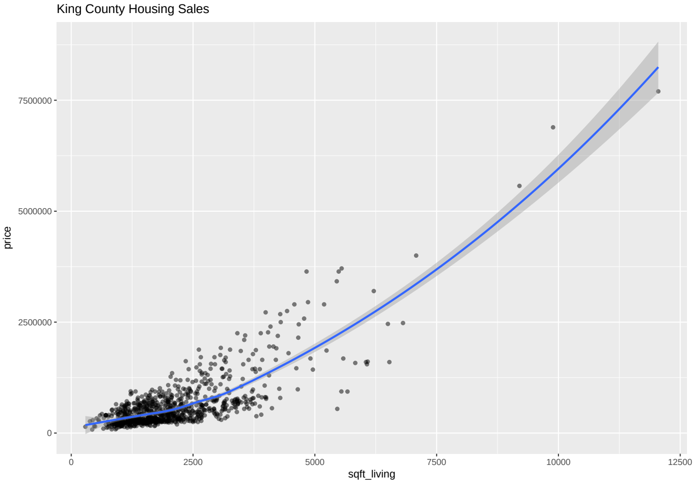
Classification Methods

Suppose we fit a model using all of the Seattle housing data, can that model be used to predict prices for homes in that data set?



Model Evaluation

We cannot assess the predictive performance by fitting a model to data and then evaluating the model using the same data.



Test / Training and Cross-Validation

There are two common options to give valid estimates of model performance:

- **Test / Training approach**. Generally 70% of the data is used to fit the model and the other 30% is held out for prediction.
- **Cross-Validation**. Cross validation breaks your data into *k* groups, or folds. Then a model is fit on the data on the *k-1* groups and then used to make predictions on data in the held out *kth group. This process continues until all groups have been held out once.

Constructing a test and training set


```
set.seed(11142017)
num.houses <- nrow(Seattle)
Seattle$zipcode <- as.factor(Seattle$zipcode)
test.ids <- base::sample(1:num.houses, size=round(num.houses*.3))
test.set <- Seattle[test.ids,]
train.set <- Seattle[(1:num.houses)[!(1:num.houses) %in%
  test.ids],]
dim(Seattle)
```

```
## [1] 869  14
```

```
dim(test.set)
```

```
## [1] 261  14
```

```
dim(train.set)
```

```
## [1] 608  14
```

Linear Regression


```r
lm.1 <- lm(price ~ bedrooms + bathrooms + sqft_living + zipcode + waterfront, data=train.set)
summary(lm.1)
```

```
##
## Call:
## lm(formula = price ~ bedrooms + bathrooms + sqft_living + zipcode +
##     waterfront, data = train.set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -929150 -116697    4505  106875 2703150
##
## Coefficients:
##                Estimate Std. Error t value             Pr(>|t|)
## (Intercept)  -163462.32   48953.91  -3.339             0.000893 ***
## bedrooms      -45354.47   14115.05  -3.213             0.001384 **
## bathrooms      -3367.16   19383.88  -0.174             0.862153
## sqft_living      341.10      16.02  21.288 < 0.0000000000000002 ***
## zipcode98014   32655.66   39869.47   0.819             0.413078
## zipcode98024  124440.49   45547.27   2.732             0.006480 **
## zipcode98032  -36965.37   40636.06  -0.910             0.363365
## zipcode98039 1275086.45   52309.52  24.376 < 0.0000000000000002 ***
## zipcode98070   99215.08   42905.03   2.312             0.021094 *
## zipcode98102  444371.35   41824.37  10.625 < 0.0000000000000002 ***
## zipcode98109  493321.89   41155.54  11.987 < 0.0000000000000002 ***
## zipcode98148   50752.12   52453.35   0.968             0.333654
## waterfront    214398.81   62381.19   3.437             0.000629 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 250600 on 595 degrees of freedom
```

Linear Regression


```
mad.lm1 <- mean(abs(test.set$price -
                        predict(lm.1,test.set)))
```

The mean absolute deviation in housing price predictions using the linear model is $162669

Polynomial Regression

Now include squared terms for square foot of living space too.

```
train.set$sqft_living2 <- train.set$sqft_living^2
test.set$sqft_living2 <- test.set$sqft_living^2

lm.2 <- lm(price ~ bedrooms + bathrooms + sqft_living + sqft_living2 +
             zipcode + waterfront, data=train.set)
summary(lm.2)
```

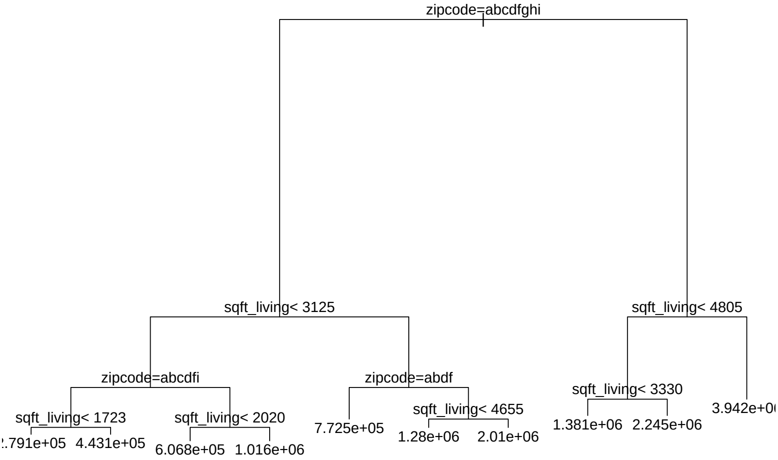
```
##
## Call:
## lm(formula = price ~ bedrooms + bathrooms + sqft_living + sqft_living2 +
##     zipcode + waterfront, data = train.set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -983741  -89990    -906   80421  929376
##
## Coefficients:
##                    Estimate     Std. Error t value            Pr(>|t|)
## (Intercept)   136424.967588   44686.362972   3.053             0.00237
## bedrooms       -2426.517180   12033.862751  -0.202             0.84027
## bathrooms       4668.853099   16132.873996   0.289             0.77238
## sqft_living        4.224443      24.590092   0.172             0.86366
## sqft_living2       0.048465       0.002973  16.302 < 0.0000000000000002
## zipcode98014   25967.543326   33169.715172   0.783             0.43402
## zipcode98024   98824.562804   37923.079692   2.606             0.00939
## zipcode98032  -75745.001595   33888.495034  -2.235             0.02578
## zipcode98039 1196173.311695   43784.387508  27.320 < 0.0000000000000002
## zipcode98070   95397.878255   35693.212367   2.673             0.00773
## zipcode98102  432516.469233   34801.046474  12.428 < 0.0000000000000002
```

Polynomial Regression


```
mad.lm2 <- mean(abs(test.set$price - predict(lm.2,test.set)))
```

Including this squared term lowers our predictive error from $162669 in the first case to $116202.

Decision Trees



Decision Trees

```
rmse.tree1 <- sqrt(mean((test.set$price - predict(tree1,test.set))^2))
mad.tree1 <- mean(abs(test.set$price - predict(tree1,test.set)))
mad.tree1
```

```
## [1] 168524.9
```

```
mad.lm1
```

```
## [1] 162668.7
```

```
mad.lm2
```

```
## [1] 116201.7
```

The predictive error for this tree, $168525 is similar to the first linear model $162669 and not quite as good as our second linear model $116202.

Ensemble Methods - Random Forest

Ensemble methods combine a large set of predictive models into a single framework. One example is a random forest - which combines a large number of trees.

While these methods are very effective in a predictive setting, it is often difficult to directly assess the impact of particular variables in the model.

Random Forest

One specific kind of ensemble method is known as a random forest, which combines several decision trees.

```
rf1 <- randomForest(price~., data=Seattle)

mad.rf <- mean(abs(test.set$price - predict(rf1,test.set)))
```

The prediction error for the random forest is substantially better than the other models we have identified $46588.

Exercise - Prediction for Capital Bike Share


```
bikes <- read.csv('http://www.math.montana.edu/ahoegh/teaching/stat408/datasets/Bike.csv')
set.seed(11142017)
num.obs <- nrow(bikes)
test.ids <- base::sample(1:num.obs, size=round(num.obs*.3))
test.bikes <- bikes[test.ids,]
train.bikes <- bikes[(1:num.obs)[!(1:num.obs) %in%
  test.ids],]
dim(bikes)
```

```
## [1] 10886    12
```

```
dim(test.bikes)
```

```
## [1] 3266   12
```

```
dim(train.bikes)
```

```
## [1] 7620   12
```

Exercise - Prediction for Capital Bike Share


```
lm.bikes <- lm(count ~ holiday + atemp,
                    data=train.bikes)
lm.mad <- mean(abs(test.bikes$count -
                        predict(lm.bikes,test.bikes)))
```

Create another predictive model and compare the results to the MAD of the linear model above (129). However, don't use casual and registered in your model as those two will sum to the total count.

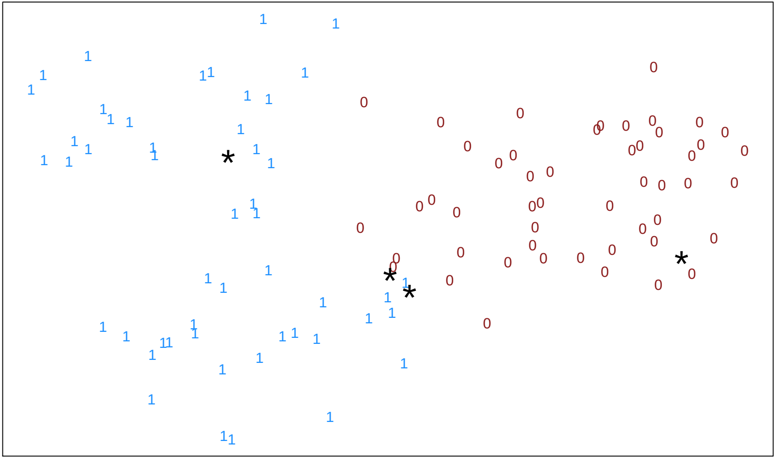
A Solution - Prediction for Capital Bike Share


```
rf.bikes <- randomForest(count ~ holiday + atemp +
     humidity + season + workingday + weather,
                data=train.bikes)
tree.mad <- mean(abs(test.bikes$count -
                  predict(rf.bikes,test.bikes)))
```

The random forest has a prediction error of (108).

Classification Methods

Classification - Given New Points (*) how do we classify them?



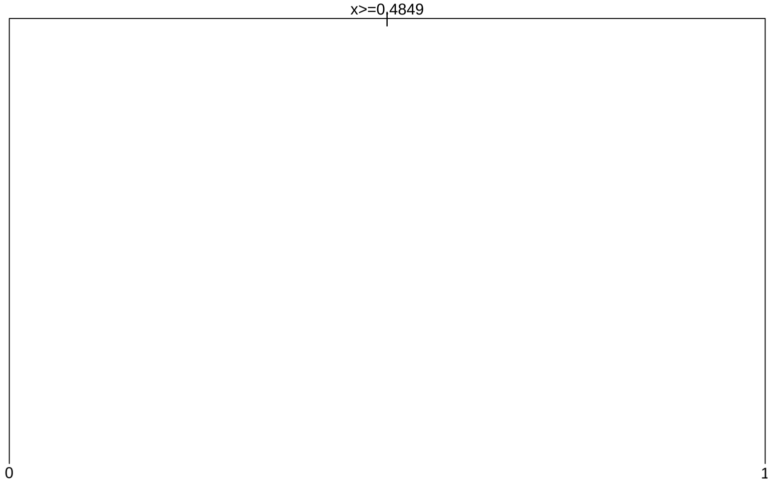
Logistic Regression


```
##
## Call:
## glm(formula = labels ~ x + y, family = "binomial", data = supervised)
##
## Deviance Residuals:
##       Min         1Q    Median         3Q        Max
## -1.93887  -0.00046   0.00000   0.00285    1.44209
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   23.898     11.813   2.023   0.0431 *
## x            -47.482     23.973  -1.981   0.0476 *
## y             -6.214      3.456  -1.798   0.0722 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 138.629  on 99  degrees of freedom
## Residual deviance:  13.024  on 97  degrees of freedom
## AIC: 19.024
##
## Number of Fisher Scoring iterations: 10
```

Logistic Regression

x	y	Prob[Val = 1]
0.20	0.70	1.000
0.45	0.35	0.588
0.48	0.30	0.319
0.90	0.40	0.000

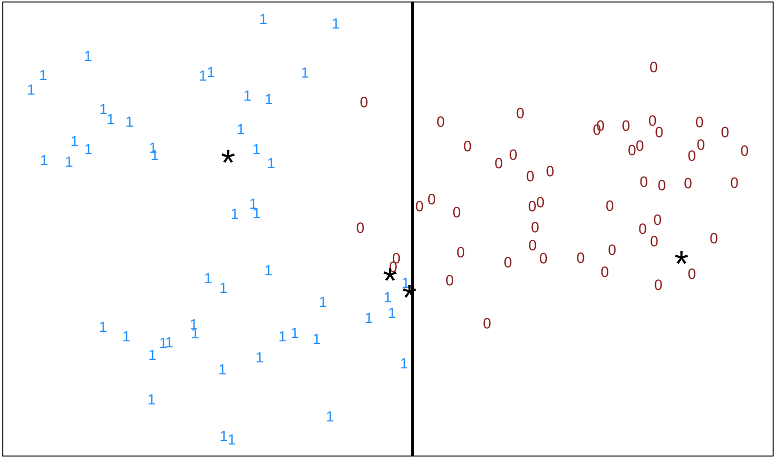
Decision Trees



Decision Trees - code


```
tree1 <- rpart(labels ~., data=supervised, method = 'class')
plot(tree1)
text(tree1)
kable(cbind(pred.points,
    round(predict(tree1,as.data.frame(pred.points))[,2],3)),
    col.names=c('x','y','Prob[Val = 1]'))
```

Decision Trees - Boundary



Exercise: Predict Titanic Survival


```
titanic <- read.csv(
  'http://www.math.montana.edu/ahoegh/teaching/stat408/datasets/titanic.
set.seed(11142017)
titanic <- titanic %>% filter(!is.na(Age))
num.pass <- nrow(titanic)
test.ids <- base::sample(1:num.pass, size=round(num.pass*.3))
test.titanic <- titanic[test.ids,]
train.titanic <- titanic[(1:num.pass)[!(1:num.pass) %in%
  test.ids],]
dim(titanic)
```

```
## [1] 714  12
```

```
dim(test.titanic)
```

```
## [1] 214  12
```

```
dim(train.titanic)
```

```
## [1] 500  12
```

Exercise: Predict Titanic Survival

See if you can improve the classification error from the model below.

```
glm.titanic <- glm(Survived ~ Age, data=train.titanic, family='binomial'
Class.Error <- mean(test.titanic$Survived != round(predict(glm.titanic,
```

The logistic regression model only using age is wrong 40% of the time.