Department of Statistics

STATS 784: Data Mining

Assignment 1 2017 Model Answer

Question 1

You got full marks if you identified two applications, gave a reasonable precis of your source, and identified your source.

Question 2

Using linear regression, construct a formula for predicting the price of a house from the other variables.

Our first step is to read in the data and think about possible data manipulations.

```
infile = "C:\\Users\\alee044\\Documents\\Teaching\\784\\Assignments\\Ass
1\\kc_house_data.csv"
housing.df = read.csv(infile, header=TRUE)
```

Comments on the variables:

1. We can ignore the ID
2. The actual date is not very important. We will extract the year and month from the date, which using the above code is read in as a string like 20141013T000000, and treat them as factors.
3. We will combine the sale date and the information on when the house was built and when (if at all) it was renovated) into a single variable age.
4. We will combine the house and basement areas into one, and create a binary variable indicating if the house has a basement.
5. We will turn year, month and view into factors.
6. It is not necessary to convert numeric binary variables into factors.
7. Zip must be as a factor.

The following code does all this and assembles the resulting variables into a new data frame.

```
# feature engineering

year = substr(as.character(housing.df$date),1,4)
month = substr(as.character(housing.df$date),5,6)
age = ifelse(housing.df$yr_renovated==0,
```

```
     as.numeric(Year) - housing.df$yr_built,
     as.numeric(Year) - housing.df$yr_renovated)
totalArea = housing.df$sqft_above + housing.df$sqft_basement
basement = (housing.df$sqft_basement==0)*1 # make it numeric

new.housing.df = data.frame(housing.df[,c(3,4,5,8,9,11,18,19,20,21)],
     view = factor(housing.df$view), year=factor(year),month=factor(month),
age=age,totalArea=totalArea,basement=basement, zip = factor(housing.df$zip))
```

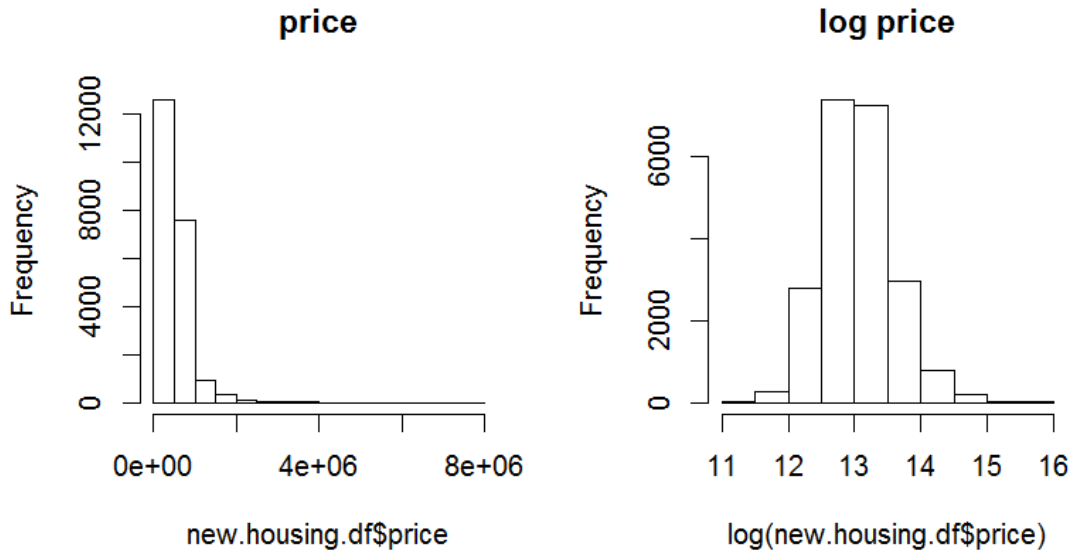
Next we check for missing values:

```
> apply(new.housing.df,2, function(x)sum(is.na(x)))
       price     bedrooms    bathrooms       floors   waterfront    condition
           0            0            0            0            0            0
         lat         long sqft_living15   sqft_lot15         view         year
           0            0            0            0            0            0
       month          age    totalArea     basement          zip
           0            0            0            0            0
```

There are none.

Finally, lets see if we should transform the response. We plot histograms of price and log price:

```
par(mfrow=c(1,2))
hist(new.housing.df$price, main = "price")
hist(log(new.housing.df$price), main = "log price")
```



Seems like a log is indicated. Next we fit the model:

```
lm.fit = lm(log(price)~., data=new.housing.df)
```

All the variables are significant (not surprising with a large sample size) so the full model will probably be the best predictor.

We will use the subset selection functions to find the subset with smallest PE.

Let's do some variable selection to see if we need all the variables:

```
null.fit = lm(log(price)~1, data=new.housing.df)
step(null.fit, formula(lm.fit), direction="forward")
step(lm.fit, formula(lm.fit), direction="back")
```

On the basis of the outputs from these functions (not shown) , it seems like the full model will be the best predictor.

Calculate estimates of prediction error for your predictor using the R330 functions cross.val and err.boot, as well as the functions crossval and bootpred in the bootstrap package, and the function train in the caret package.

How accurate do you think your estimates are?

```
# measurement of PE
> library(R330)
> library(bootstrap)
> library(caret)
```

First, cross-validation:

```
> # R330 functions
> cross.val(lm.fit, nfold=10, nrep=20)
Cross-validated estimate of root
mean square prediction error =  0.1948888
> cross.val(lm.fit, nfold=5, nrep=20)
Cross-validated estimate of root
mean square prediction error =  0.194966


> caret functions
> train(log(price)~., data=new.housing.df,
 method = "lm", # you get this from the website
 trControl = trainControl(method="cv", number=10,
 repeats=20))

Resampling results

  RMSE       Rsquared   RMSE SD      Rsquared SD
  0.1947732  0.8633441  0.004343356  0.007098211

train(log(price)~., data=new.housing.df,
```

```
 method = "lm", # you get this from the website
 trControl = trainControl(method="cv", number=5,
 repeats=20))
Resampling results

  RMSE       Rsquared   RMSE SD      Rsquared SD
  0.1950162  0.8630575  0.004754068  0.003237339

# boostrap functions
theta.fit <- function(x,y){lsfit(x,y)}
theta.predict <- function(fit,x){cbind(1,x)%*%fit$coef}
sq.err <- function(y,yhat) { (y-yhat)^2}

y =log(new.housing.df[,1])
x = model.matrix(lm.fit)[,-1] # see comment below

# repeat 20 times
N=20
cv10Vec = numeric(N)
for(i in (1:N)){
  cv10Errs = crossval(x,y, theta.fit, theta.predict,ngroup=10)
  cv10Vec[i] = mean((y-cv10Errs$cv.fit)^2)
}
mean(cv10Vec)
[1] 0.03798151

# repeat 20 times
N=20
cv5Vec = numeric(N)
for(i in (1:N)){
  cv5Errs = crossval(x,y, theta.fit, theta.predict,ngroup=5)
  cv5Vec[i] = mean((y-cv5Errs$cv.fit)^2)
}
mean(cv5Vec)
[1] 0.03799834
```

Now the bootstrap

```
#  bootstrap with 50 reps

# R330

err.boot(lm.fit, B=50)

$err
[1] 0.0375622

$Err
[1] 0.03820315
```

```
# caret

train(log(price)~., data=new.housing.df,
 method = "lm", # you get this from the website
 trControl = trainControl(method="boot", repeats=50))

Resampling results

  RMSE       Rsquared   RMSE SD      Rsquared SD
  0.1954108  0.8620392  0.002740157  0.004396371


train(log(price)~., data=new.housing.df,
 method = "lm", # you get this from the website
 trControl = trainControl(method="boot632", repeats=50))

Resampling results

  Rsquared SD
  0.194636  0.8638792  0.002064829  0.003101573

# Bootstrap package

> boot = bootpred(x,y,nboot=50, theta.fit, theta.predict,
+  err.meas=sq.err)


> boot
[[1]]
[1] 0.0375622

[[2]]
[1] 0.0004536646

[[3]]
[1] 0.03793542
```

How accurate do you think your estimates are?

The table below gives a summary of the results:

	R330	bootstrap	caret
CV5	0.03801	0.03700	0.03803
CV10	0.03798	0.03798	0.03793
Err+opt	0.03820	0.03801	0.03819
632	-	0.03793	0.03788

The results are very consistent, an estimated prediction error of around 0.038.

Points to note:

1. The bootpred function in the bootstrap package is very slow.

2. The crossval function in the bootstrap package is a bit tricky to use: the input x has to be a numeric matrix of all the variables, including dummy variables generated by the factors, but excluding the column if 1`s corresponding to the intercept. It can be calculated using the model.matrix function, but you have to remove the column of ones.

3. With such a big sample size (relative to the number of variables) the training error 0.0375622 is not too optimistic.

-