eBuy

author: S. Jackson Kelley (A01281942) date: November 16, 2016 autosize: true

Problem Description

Given past sold items, can we predict the price of future items?

Tools

- JavaScript
- eBay API
- R
- mongoDB
- AWS

Methods

- Machine Learning
- Cross Validation
- Data Science
- Text Analysis
- Data Wrangling

Code Bases

- ebay-api open source javascript library (for data retrieval, shown in previous presentation)
- Hmisc (for data imputation of NA's in R data frame)
- ggplot2/ggmap (for visualization of data set in R)
- tm/wordcloud (title analysis and visualization library)
- randomForest (for variable importance and as candidate ML algorithm)
- caret (for cross validation with ML algorithms and ML packages)

Progress

Currently I've stored eBay data for roughly 2 weeks (between the last presentation and now) of Playstation 4s in an AWS server running the JavaScript code everyday

and storing this data into a mongoDB database.

In my previous presentation feature selection was flagged as a potential challenge with this project. Therefore the mongoDB data has been pulled out of the AWS server and into R for statistical analysis and visualization in order to make statistically informed decisions in regard to which features will be used in the various machine learning algorithms.

{r, echo=FALSE} library(Hmisc) library(ggplot2)

Progress

Various pre-processing steps performed to clean the data for analysis and machine learning training.

```
outcomes = read.csv("completedItems.csv",header=TRUE)
forSale = read.csv("forSale.csv",header=TRUE)
#data imputation for NA's in dataset.
forSale$shippingCost[forSale$calculateShipping == "true"] = NA
forSale$shippingCost = with (forSale,impute(shippingCost,mean))
forSale$bidCount     = with(forSale, impute(bidCount,mean))
#convert bidCount to a numeric value
forSale$bidCount = as.numeric(forSale$bidCount)
forSale$totalPrice = forSale$currentPrice + forSale$shippingCost
forSale$totalPrice = as.numeric(forSale$totalPrice)
# remove all non Video Game Console categories from the outcome dataset.
outcomes = outcomes[outcomes$categoryName == "Video Game Consoles",]
dates = as.Date(forSale$endDate)
```

Progress

(pre-processing continued)

```
# convert dates into a day of the week categorical variable
forSale$dayOfWeek = as.factor(weekdays(dates))
# remove outliers from the forSale dataset
outliers = boxplot.stats(forSale$totalPrice)$out
forSale = forSale[!forSale$totalPrice %in% outliers,]
forSale = forSale[(forSale$totalPrice > 100),]
#merge the forSale dataset with the final selling price dataset
fullDataset <- merge(forSale,outcomes,by="X_id")
```

Much more, but I won't bore you with the details.

Progress

Summary plots generated by R of the distribution of prices for the dataset.

```
ggplot(outcomes, aes(sellingPrice, fill = categoryName)) +
  geom_histogram(binwidth = 5) + labs(title = "All Sold Playstation 4s")
ggplot(forSale, aes(totalPrice, fill = listingType)) +
  geom_histogram(binwidth = 5)+ labs(title = "Playstation 4s For Sale in the Last Two Weeks'
```

Progress

Summary plots generated by R visualizing whether the day of the week or "buy it now" is statistically significant to price sold.

```
ggplot(fullDataset, aes(y = sellingPrice, x = factor(dayOfWeek))) +scale_x_discrete("Day Of
        scale_y_continuous("Final Selling Price") + geom_boxplot(outlier.color="red") + labs

averagePricePerBuyItNow = with(fullDataset, tapply(sellingPrice, buyItNowAvailable,mean))
plotDF = aggregate(fullDataset$sellingPrice ~ fullDataset$buyItNowAvailable, FUN = mean)
ggplot(fullDataset, aes(y = sellingPrice, x = factor(buyItNowAvailable))) +scale_x_discrete(
        scale_y_continuous("Final Selling Price") + geom_boxplot(outlier.color="red")+ labs(
```

Progress

Summary plots generated by R visualizing whether there is correlation to the price items are sold for and the number of bids for various listing types.

```
p1 <- ggplot(fullDataset, aes(x = bidCount, y = sellingPrice, color = factor(conditionDispla
p1
```

Progress

A Google map with sold Playstation 4's plotted on it.

Progress

A word map showing the most dense words in all item titles using tf-idf.

```
library(tm)
library(wordcloud)
text = Corpus(VectorSource(fullDataset$title))
```

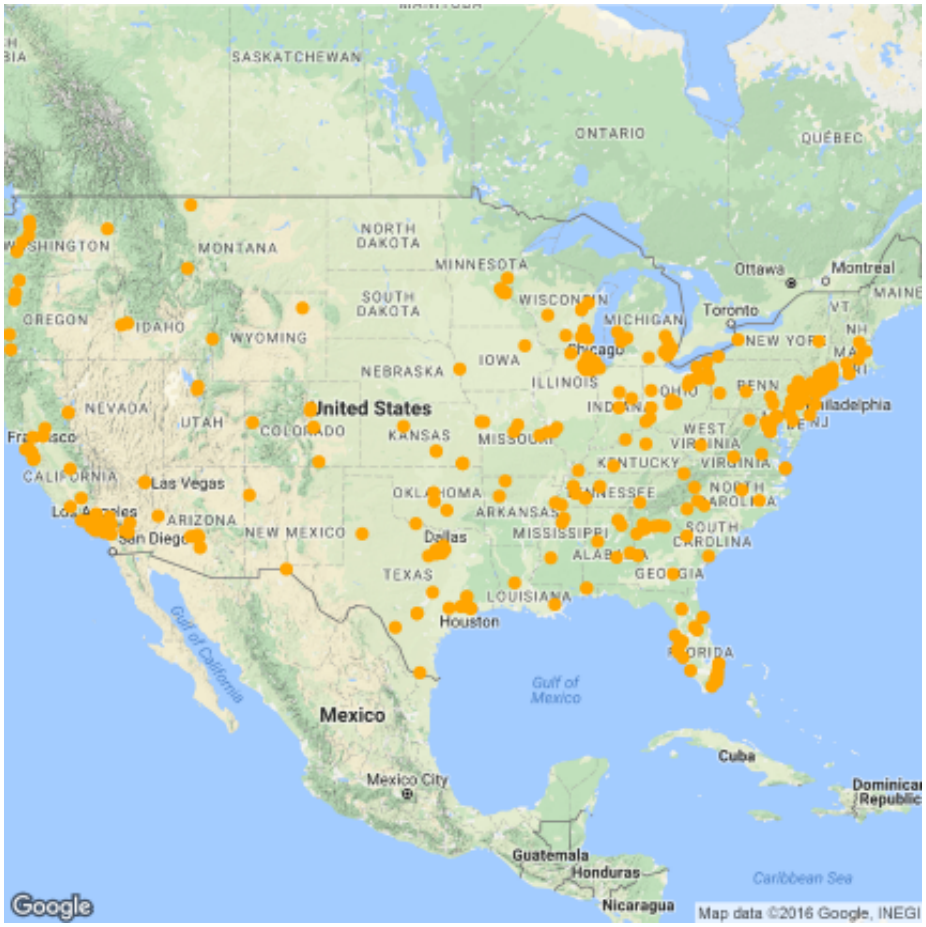
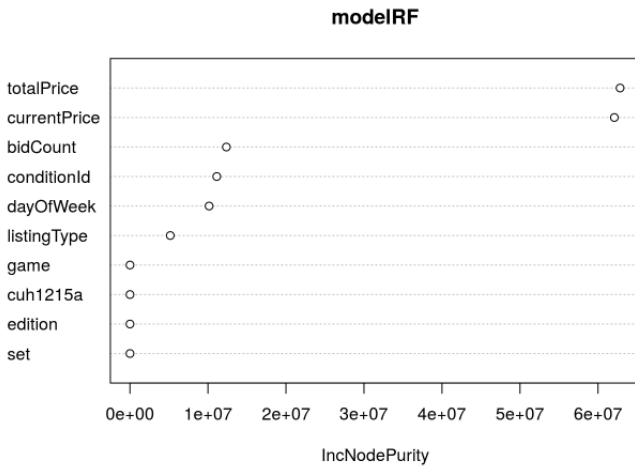


Figure 1: title

```
text = tm_map(text,removePunctuation)
text = tm_map(text,content_transformer(tolower))
text = tm_map(text,removeWords,stopwords("english"))
text <- tm_map(text, stripWhitespace)
dtm = DocumentTermMatrix(text)
freq = colSums(as.matrix(dtm))
ord = order(freq)
dtms = removeSparseTerms(dtm,.9)
set.seed(142)
wordcloud(names(freq),scale=c(8,.5),freq,max.words=100,rot.per=.2,colors=brewer.pal(6,"Dark2
```

Progress



Variable importance plot from Random Forest model.

Plans

The data has been cleaned and analyzed, and predictors have been selected as important by the Random Forest algorithm as well as various hypothesis tests (not shown in this presentation). The next step will be training different machine learning algorithms and reporting on their accuracy.

After the semester ends I plan on implementing the selected machine learning algorithm on-line (I believe using Spark, but I have no experience with big data frameworks) to execute predictions in real time. I will then integrate this feature into the existing software and test whether the machine learned model is profitable in comparison with the current model.

Expected Deliverables

- A selected machine learning algorithm
- A measure of accuracy for the selected algorithm
- A list of candidate items for purchase from the dataset
- A predicted price associated with the items
- Expected profit from selected items