Main

Vikas Arun(va2298), Xuehan Liu(xl2615), Sean Reddy(sr3336), Boxuan Zhao(bz2290)

4/28/2017

Introduction

In this project, we look at auction items' price on sothebys.com and focus on trying to predict the sale price of paintings auctioned on the website. Inspiration for the project comes for the inaccuracy of the predictions presented on Sotheby's site. In order to predict the price we perform text mining on all the text describing the painting, as well as image analysis on the image of the painting itself. We first scrape information, including descriptions and images of each items, and then do cleaning-up and image feature extraction. After obtaining all the features, we fit a model and fit it to our dataset with features.

Step 0: Install and Import Relevant Packages

Changed the working directory to the /doc folder on your local machine.

```
#Package needed
list.of.packages = c("glmnet","Hmisc","jpeg","OpenImageR","EBImage")

new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]

if(length(new.packages))
{
  install.packages(new.packages)
  source("http://bioconductor.org/biocLite.R")
  biocLite("EBImage")
}

suppressMessages(library(glmnet))
suppressMessages(library(Hmisc))
suppressMessages(library(EBImage))
suppressMessages(library(jpeg))
suppressMessages(library(OpenImageR))


setwd("/Users/xuehan/Desktop/Spr2017-proj5-grp8/lib") #Please change to your local path
```

Step 1: Pull data from sothebys.com

The Sothebys web scraping program was written in Python and can be accessed through "GoldDust_1.0.ipynb", and viewing the file through a Jupyter Notebook. The information was scraped by going to each auction page, downloading the item names, and then navigating to each item page to download the title, image, description, sale price, predicted price, and many other features. These items were outputted to two places. The text data was outputted to "auctionItems.csv" in the /data folder and the images were outputted to "Images_full", also in the /data folder. We scraped ~3000 painting items from Sothebys, however Sotheby's classification of auctions isn't perfect, so the data didn't contain solely paintings.

Step 2: Feature clean up and extractions

We scraped information from Sotheby's website and the initial features we have from the csv file are the following:

"auctionDate","auctionId","auctionName","currency","genre", "guaranteeLine","highEst","id","isSold","lowEst","medium","sa
"auctionYear", "lot_desc"

Binary Categorical Variables

We created categorical variables(1s and 0s) "History", "Portrait", "Landscape", "Genre", "StillLife", Abstract, and "Other" from the column "type", in which we utilized regrular expressions to extract type for each paintings. In addition we created a variable called "Animal", which indicates whether the title or description contains any reference to a major animal. Finally, we created the variables "famous"" and "sold"", which record whether each painting is owned by famous people and whether it is sold, based on the information we extract from lot_desc and isSold

Continuous Variables

We created diff variable, which is estimated by the increased proportion from highEst and lowEst. Finally, we extract the height and width of the paintings from the lot_desc and form the final two features from our data set.

Image Variables

In addition to the csv file, we have also created HOG feature as well as RGB feature from the paintings.

Step 2a: Data clean-up and feature extraction

```
source("../lib/FeatureConstruction.R")
my.dat = read.csv("../data/auctionItems.csv",header = TRUE, stringsAsFactors = FALSE)
feature.csv = Feature.Construction(my.dat)#Construct the first type of features
#dim(feature.csv)
animal<-read.csv("../output/AnimalFeature.csv")
colnames(animal)<-c("ID","animal")

#In the following step, we use the images' captions to rename the observations' ID
#Notice that the images we used are too large too upload on Gitgub. Alternatively, all 3000 images are

#Rename ID for each observations in the dataset
img_dir_full <- "/Users/xuehan/Desktop/Images/" #change to your local path of 3000 images
file.names <- list.files(img_dir_full,pattern="*.jpg")
file.names.short<-NULL
for (i in 1:length(file.names)){
  file.names.short[i]<-substr(file.names[[i]],1,nchar(file.names[[i]])-4)
}

for (i in 1:nrow(animal)){
  animal$ID[i]<-file.names.short[i]
}
```

```
#Merge two datasets
data.full1<-merge(feature.csv,animal,by.x="ID",by.y="ID")
```

Step 2b: Images feature extraction

In this step, we extracted HoG, RGB and VGG feature from the images that associate with each auction item. Since most of our observations are paintings, it makes sense that we extract information from those paintings directly.

Notice that the images we used are too large too upload on Gitgub. Alternatively, all 3000 images are in the Google drive (https://drive.google.com/drive/folders/0BzExzKbmVUJ2R3dsYUVYUXNvMVk?usp=sharing). Images are stored seperated in three different files (image_2500,image_5000, and image_7000). If you are interested in reproduce this project, please download them and move them all in the same folder for future use. Anyone with a LionMail should be able to access the google drive folder. Finally, please note that VGG features can be extracted using the "VGG Model.ipynb" file and opening it with Jupyter Notebook. The VGG model is a pretrained model from the tensorflow package, also installed under the /lib folder. These features, while extractable using the above code, were not used in the final algorithm since VGG takes ~10 min per image to extract features.

```
#After downloading the images to your local path, please go to the "image_feature_extraction.R" and cha
#Then run the following code


#source("../lib/image_feature_extraction.R")



#The above file will take approximately 3 hours to run. Basically, it will generate two csv files for H

#You can manually run rgb_seperate.R under doc folder. It generates the whole 3000 by 512 matrix in fou


###HoG Feature

HoG<-read.csv("../data/HoG.csv",header=T)[,-1]

#head(HoG)


###RGB Feature
RGB<-read.csv("../data/RGB.csv",header=T)[,-1]

#head(RGB)
```

Step 2c: Merge all sub-datasets that contain features together to generate a functional dataset with full features for model fitting.

The dataset we created does not have the sale price, so we merge our dataset back with the original file in order to retrived the sale price. We use the auction ID and item ID combination as the key to join the datasets.

```
#Merge datasets by paintings' ID which is the last column
head(HoG[,ncol(HoG)])
```

```
## [1] american-art-n09633_1   american-art-n09633_10  american-art-n09633_100
```

```
## [4] american-art-n09633_101 american-art-n09633_102 american-art-n09633_103
## 3008 Levels: american-art-n09633_1 ... so-contemporary-n08929_68
```

```
colnames(HoG)[ncol(HoG)]<-"ID"
head(RGB[,ncol(RGB)])
```

```
## [1] american-art-n09633_1   american-art-n09633_10  american-art-n09633_100
## [4] american-art-n09633_101 american-art-n09633_102 american-art-n09633_103
## 3008 Levels: american-art-n09633_1 ... so-contemporary-n08929_68
```

```
colnames(RGB)[ncol(RGB)]<-"ID"

data.full<-merge(HoG,RGB,by.x="ID",by.y="ID")
colnames(data.full)[2:55]<-paste("HoG",1:54,sep="")
colnames(data.full)[56:567]<-paste("RGB",1:512,sep="")
#store full data
data_full<-merge(data.full,data.full1,by.x="ID",by.y="ID" )
write.csv(data_full,file="../data/data_full.csv")
```

Step 3: Fit the model and find those important features in order to predict auction price

For this part, we first read in and clean the data, then we apply the regular linear regression method, which result in a non-satifiactory result. The reason it does not result in a satisfactory result is that the vast majority of the features are not useful in predicting the y variable, and simply add variance. Therefore, we then implement Lasso regression and achieve a much better results.

```
#read in the cleaned up data
data_full<-read.csv("../data/data_full.csv",header = TRUE, stringsAsFactors = FALSE)
data<-read.csv("../data/auctionItems.csv",header = TRUE, stringsAsFactors = FALSE)[,c(3,9,13)]
data$Id<-paste0(data$auctionId,"_",data$id)
data<-data[,-c(1,2)]
data_full<-merge(data_full,data,by.x="ID",by.y="Id")
colnames(data_full)[3:56]<-paste("HoG",1:54,sep="")
colnames(data_full)[57:568]<-paste("RGB",1:512,sep="")
data_full = data_full[,-c(1,2)]

####Regular regression methods:
#We randomly split 25% of our data as testing data set:
ind<-sample(1:dim(data_full)[1],dim(data_full)[1] * 0.25)
test<-data_full[ind,]
train<-data_full[-ind,]
score<- lm(salePrice~.,train)#regular regression
regular.prediction = predict(score,test[,-c(580)])#Store predictions from regular regression model
na.position = which(is.na(regular.prediction))#Locate na positions
regular.prediction = regular.prediction[-na.position]#Remove all na predicitons from regular regression

#Produce statistics for comparison
regular.summary = summary(regular.prediction-as.numeric(test$salePrice)[-na.position])
regular.mse = sum((regular.prediction-as.numeric(test$salePrice)[-na.position])^2)

####Advanced regression methods : LASSO Regression
#We randomly split 25% of our data as testing data set:
data_omit = na.omit(data_full)#Remove any rows that has na valuse in it
data_omit$salePrice = as.numeric(data_omit$salePrice)#Convert the character into numerical value
ind<-sample(1:dim(data_omit)[1],dim(data_omit)[1]*0.25)
```

```
test<-data_omit[ind,]
train<-data_omit[-ind,]
lambda = cv.glmnet(x=as.matrix(train[,1:579]),y=train[,580],type.measure = "mse")$lambda.min#Cross vali
score_lasso = glmnet(as.matrix(train[,1:579]),as.matrix(train[,580]),family="gaussian",alpha = 1, lambda
lasso.prediction = predict(score_lasso,as.matrix(test[,-c(580)]))#Prodce prediction

#Produce statistics for comparison
lasso.summary = summary(lasso.prediction - as.numeric(test[,580]))
lasso.mse = mean((lasso.prediction-test[,c(580)])^2)
```

Compare statistics summary for two models

```
#mse
regular.mse
```

```
## [1] 5.079951e+14
```

```
lasso.mse
```

```
## [1] 1662517565
```

```
#summary
regular.summary
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -19050000    -14260      5065     -9675     21840   6034000
```

```
lasso.summary
```

```
##        s0
##  Min.   :-528483
##  1st Qu.:   2602
##  Median :  11008
##  Mean   :   4374
##  3rd Qu.:  16309
##  Max.   : 441731
```

The MSE above of regular regression get reduced almost

Based on the statistics summary above, the lasso regression is much better than the regular regression. The "best" model for predicting the auction price should be the lasso regression.

You can see that the Mean/Median residual is much more reasonable for the lasso model than the standard linear regression model. You will also notice that the MSE is much lower for lasso than standard linear regression. This is useful both from a prediction point of view but also from an interpretability point of view, as it allows us to focus on a small number of features that are useful in predicting painting prices, and extrpolate insight from those features.