Project 3 - Main Script

Group 12: Vikas Arun, Xuanyu Xiao, Boxuan Zhao, Boya Zhao, Yuxi Zhou

Revlevant packages needed for this file

```
list.of.packages <- c("e1071", "ggplot2","gbm","caret","randomForest","EBImage")

new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages))
  {
   install.packages(new.packages)
   source("https://bioconductor.org/biocLite.R")
   biocLite("EBImage")
  }

library("gbm")
library("ggplot2")
library("caret")
library("randomForest")
library("EBImage")
```

Step 1: specify directories.

This directory should be set to the lib folder of the cloned repository

```
knitr::opts_knit$set(root.dir = "../lib")
knitr::opts_knit$set(width = 150)
# here replace it with your own path or manually set it in RStudio to where this rmd file is located.
```

Providing directories for images, sift features, and labels. Providing paths for ouputted models and predictions.

```
#image_test_dir <- "../data/test_data/raw_images" # This will be modified for different data sets.
#image_train_dir <- "../data/train_data/raw_images"
#img_train_dir <- paste(experiment_dir, "train/", sep="")
#img_test_dir <- paste(experiment_dir, "test/", sep="")
image_all.dir <- "../data/training_data/raw_images"
original_data_train = "../data/sift_ori_train.csv"
original_data_test = "../data/sift_ori_test.csv"
modified_data_train = "../data/sift_simp_gray_train.csv"
modified_data_test = "../data/sift_simp_gray_test.csv"
labels_train = "../data/labels_train.csv"
labels_test = "../data/labels_test.csv"

gbm_model_original_features = "../output/GBMFullFeature.RData"
rf_model_original_features = "../output/RFFullFeature.RData"
gbm_model_modified_features = "../output/GBMModifiedFeature.RData"
rf_model_modified_features = "../output/RFModifiedFeature.RData"

gbm_model_original_predict = "../output/GBMFullFeaturePredictions.csv"
rf_model_original_predict = "../output/RFFullFeaturePredictions.csv"
gbm_model_modified_predict = "../output/GBMModifiedPredictions.csv"
rf_model_modified_predict = "../output/RFModifiedPredictions.csv"
```

Step 2: set up controls for evaluation experiments.

In this chunk, ,we have a set of controls for the evaluation experiments.

- (T/F) cross-validation on the training set for GBM

- (number) K, the number of CV folds
- (T/F) Out of Bag Estimate (similar to cross-validation) on training set for Random Forest
- (T/F) process features for training set
- (T/F) run evaluation on an independent test set

```
run.cv=FALSE # run cross-validation on the training set
K <- 5  # number of CV folds
run.OOB=FALSE
run.feature.train=TRUE # process features for all pictures
run.test=TRUE # run evaluation on an independent test set
#run.feature.test=TRUE # process features for test set
```

Using cross-validation or independent test set evaluation, we compare the performance of different classifiers or classifiers with different specifications. For the GBM model, shrinkage values of .001, .01, and .1 are evaluted, as well as a size limit of 100, 500, and 1000 trees. For the Random Forest model, a size limit of 100, 500, and 1000 trees is evaluted using the Out of Bag (OOB) error estimate, which is similar to cross validation.

Step 3: construct visual feature for Full images

Features are created by doing two things. *First, the number of provided sift features is reduced. Sift feature with standard deviation in the lowest 25th percentile are thrown out. Additionaly for each feature, the mean value for "chicken" images is subtracted from the mean value for "poodle" images. Features, with the absolute value of differences less than the median are discarded.* Second, grayscale features are added. For each image, a frequency histogram is created, representing the percentage of pixels falling in each of 256 gray scale bins. As such, each image has 256 grayscale features added.

The data is also split into training and testing data in a 75/25 split.

```
source("../lib/feature.R")
source("../lib/DataSplit.R")

tm_feature <- NA
if(run.feature.train){
  #tm_feature <- system.time({
   # dat_all <- feature(img_dir=image_all.dir)})
  #tm_feature <-
    feature(img_dir=image_all.dir)
                                     #feature(img_train_dir,
                                     #                  "train",
                                      #                 data_name="zip",
                                       #                export=TRUE))
}
```

```
## Elapsed training time for featurizer is  681.786  seconds
```

```
#SPlit the data in to train and test sets

dataSplit.cv()

#tm_feature_train <- NA
#if(run.feature.train){
 # tm_feature_train <- system.time({
  #  dat_train <- feature(img_dir=image_train_dir)})
                                    #feature(img_train_dir,
                                     #                  "train",
                                      #                 data_name="zip",
                                       #                export=TRUE))
#}




#tm_feature_test <- NA
```

```
#if(run.feature.test){
 # tm_feature_test <- system.time(dat_test <- feature(img_test_dir,
  #                                                   "test",
   #                                                  data_name="zip",
    #                                                 export=TRUE))
#}

#write(dat_all,file="../output/feature_all.csv")
#save(dat_train, file="./output/feature_train.RData")

#save(dat_test, file="./output/feature_test.RData")
```

Step 4: Model Training and Parameter Selection

Training the GBM model and Random Forest model on the original features and the new features. Outputed models are stored in RData files in the output folder. Cross validation and OOB parameter estimates are done if requested.

```
source("../lib/train.R")
source("../lib/test.R")
```

```
train_models(original_data_train, labels_train, full_feature = TRUE, run_cv = run.cv, run_OOB = run.OOB, K = K
```

```
## Loading required package: plyr

## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3732             nan     0.1000    0.0042
##      2        1.3613             nan     0.1000    0.0052
##      3        1.3485             nan     0.1000    0.0051
##      4        1.3379             nan     0.1000    0.0040
##      5        1.3290             nan     0.1000    0.0030
##      6        1.3199             nan     0.1000    0.0031
##      7        1.3118             nan     0.1000    0.0024
##      8        1.3032             nan     0.1000    0.0036
##      9        1.2951             nan     0.1000    0.0026
##     10        1.2872             nan     0.1000    0.0022
##     20        1.2215             nan     0.1000    0.0020
##     40        1.1300             nan     0.1000    0.0006
##     60        1.0625             nan     0.1000   -0.0012
##     80        1.0041             nan     0.1000   -0.0000
##    100        0.9598             nan     0.1000    0.0003
##    120        0.9154             nan     0.1000   -0.0007
##    140        0.8761             nan     0.1000   -0.0001
##    160        0.8368             nan     0.1000    0.0001
##    180        0.8022             nan     0.1000    0.0001
##    200        0.7735             nan     0.1000   -0.0010
##    220        0.7433             nan     0.1000   -0.0002
##    240        0.7129             nan     0.1000   -0.0004
##    260        0.6847             nan     0.1000   -0.0000
##    280        0.6595             nan     0.1000   -0.0001
##    300        0.6320             nan     0.1000    0.0001
##    320        0.6090             nan     0.1000   -0.0002
##    340        0.5885             nan     0.1000   -0.0000
##    360        0.5678             nan     0.1000   -0.0001
##    380        0.5463             nan     0.1000   -0.0004
##    400        0.5256             nan     0.1000   -0.0001
##    420        0.5076             nan     0.1000   -0.0005
##    440        0.4901             nan     0.1000   -0.0000
##    460        0.4744             nan     0.1000   -0.0002
##    480        0.4592             nan     0.1000   -0.0002
##    500        0.4425             nan     0.1000   -0.0003
```

```
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3737             nan     0.1000    0.0061
##      2        1.3634             nan     0.1000    0.0038
##      3        1.3530             nan     0.1000    0.0035
##      4        1.3395             nan     0.1000    0.0039
##      5        1.3286             nan     0.1000    0.0041
##      6        1.3179             nan     0.1000    0.0025
##      7        1.3077             nan     0.1000    0.0038
##      8        1.2969             nan     0.1000    0.0047
##      9        1.2902             nan     0.1000    0.0024
##     10        1.2804             nan     0.1000    0.0032
##     20        1.2057             nan     0.1000    0.0015
##     40        1.1038             nan     0.1000    0.0000
##     60        1.0293             nan     0.1000   -0.0002
##     80        0.9686             nan     0.1000   -0.0000
##    100        0.9194             nan     0.1000   -0.0009
##    120        0.8742             nan     0.1000   -0.0002
##    140        0.8373             nan     0.1000   -0.0018
##    160        0.7990             nan     0.1000   -0.0003
##    180        0.7656             nan     0.1000   -0.0001
##    200        0.7359             nan     0.1000   -0.0000
##    220        0.7029             nan     0.1000   -0.0001
##    240        0.6747             nan     0.1000   -0.0001
##    260        0.6500             nan     0.1000    0.0002
##    280        0.6257             nan     0.1000   -0.0005
##    300        0.6033             nan     0.1000   -0.0006
##    320        0.5796             nan     0.1000    0.0000
##    340        0.5584             nan     0.1000   -0.0004
##    360        0.5390             nan     0.1000   -0.0001
##    380        0.5179             nan     0.1000   -0.0001
##    400        0.4989             nan     0.1000   -0.0001
##    420        0.4820             nan     0.1000   -0.0002
##    440        0.4657             nan     0.1000   -0.0001
##    460        0.4500             nan     0.1000   -0.0000
##    480        0.4334             nan     0.1000   -0.0001
##    500        0.4180             nan     0.1000   -0.0006
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3725             nan     0.1000    0.0044
##      2        1.3632             nan     0.1000    0.0028
##      3        1.3511             nan     0.1000    0.0050
##      4        1.3408             nan     0.1000    0.0044
##      5        1.3291             nan     0.1000    0.0046
##      6        1.3204             nan     0.1000    0.0028
##      7        1.3133             nan     0.1000    0.0016
##      8        1.3032             nan     0.1000    0.0038
##      9        1.2928             nan     0.1000    0.0047
##     10        1.2835             nan     0.1000    0.0029
##     20        1.2133             nan     0.1000    0.0024
##     40        1.1237             nan     0.1000    0.0014
##     60        1.0526             nan     0.1000    0.0010
##     80        0.9929             nan     0.1000   -0.0004
##    100        0.9443             nan     0.1000   -0.0008
##    120        0.8963             nan     0.1000   -0.0002
##    140        0.8555             nan     0.1000   -0.0005
##    160        0.8204             nan     0.1000   -0.0005
##    180        0.7864             nan     0.1000    0.0002
##    200        0.7537             nan     0.1000   -0.0002
##    220        0.7233             nan     0.1000   -0.0005
```

```
##    240        0.6949             nan     0.1000   -0.0000
##    260        0.6667             nan     0.1000   -0.0005
##    280        0.6423             nan     0.1000   -0.0003
##    300        0.6159             nan     0.1000   -0.0004
##    320        0.5947             nan     0.1000   -0.0005
##    340        0.5722             nan     0.1000   -0.0005
##    360        0.5496             nan     0.1000    0.0002
##    380        0.5295             nan     0.1000   -0.0007
##    400        0.5128             nan     0.1000   -0.0004
##    420        0.4952             nan     0.1000   -0.0004
##    440        0.4789             nan     0.1000   -0.0002
##    460        0.4623             nan     0.1000   -0.0002
##    480        0.4468             nan     0.1000   -0.0002
##    500        0.4308             nan     0.1000   -0.0002
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3754             nan     0.1000    0.0027
##      2        1.3650             nan     0.1000    0.0035
##      3        1.3538             nan     0.1000    0.0045
##      4        1.3415             nan     0.1000    0.0052
##      5        1.3302             nan     0.1000    0.0047
##      6        1.3190             nan     0.1000    0.0038
##      7        1.3096             nan     0.1000    0.0035
##      8        1.3001             nan     0.1000    0.0037
##      9        1.2919             nan     0.1000    0.0034
##     10        1.2843             nan     0.1000    0.0025
##     20        1.2182             nan     0.1000    0.0014
##     40        1.1292             nan     0.1000   -0.0008
##     60        1.0567             nan     0.1000    0.0007
##     80        0.9967             nan     0.1000    0.0011
##    100        0.9456             nan     0.1000    0.0002
##    120        0.9022             nan     0.1000   -0.0007
##    140        0.8627             nan     0.1000    0.0007
##    160        0.8262             nan     0.1000   -0.0002
##    180        0.7891             nan     0.1000   -0.0012
##    200        0.7557             nan     0.1000   -0.0001
##    220        0.7251             nan     0.1000   -0.0001
##    240        0.6964             nan     0.1000    0.0000
##    260        0.6684             nan     0.1000   -0.0002
##    280        0.6428             nan     0.1000   -0.0007
##    300        0.6190             nan     0.1000   -0.0003
##    320        0.5976             nan     0.1000   -0.0000
##    340        0.5756             nan     0.1000    0.0001
##    360        0.5559             nan     0.1000   -0.0001
##    380        0.5362             nan     0.1000   -0.0005
##    400        0.5187             nan     0.1000   -0.0003
##    420        0.4990             nan     0.1000   -0.0003
##    440        0.4814             nan     0.1000   -0.0006
##    460        0.4654             nan     0.1000   -0.0009
##    480        0.4489             nan     0.1000   -0.0002
##    500        0.4332             nan     0.1000   -0.0002
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3732             nan     0.1000    0.0042
##      2        1.3606             nan     0.1000    0.0043
##      3        1.3502             nan     0.1000    0.0045
##      4        1.3420             nan     0.1000    0.0021
##      5        1.3305             nan     0.1000    0.0049
##      6        1.3193             nan     0.1000    0.0036
##      7        1.3106             nan     0.1000    0.0026
```

```
##      8        1.3008             nan     0.1000    0.0035
##      9        1.2930             nan     0.1000    0.0026
##     10        1.2844             nan     0.1000    0.0020
##     20        1.2145             nan     0.1000    0.0017
##     40        1.1197             nan     0.1000    0.0005
##     60        1.0503             nan     0.1000    0.0008
##     80        0.9931             nan     0.1000    0.0004
##    100        0.9439             nan     0.1000   -0.0001
##    120        0.9014             nan     0.1000   -0.0006
##    140        0.8624             nan     0.1000   -0.0002
##    160        0.8227             nan     0.1000   -0.0004
##    180        0.7913             nan     0.1000   -0.0004
##    200        0.7610             nan     0.1000   -0.0001
##    220        0.7299             nan     0.1000   -0.0002
##    240        0.7026             nan     0.1000   -0.0002
##    260        0.6763             nan     0.1000   -0.0003
##    280        0.6520             nan     0.1000   -0.0006
##    300        0.6283             nan     0.1000   -0.0006
##    320        0.6039             nan     0.1000   -0.0001
##    340        0.5825             nan     0.1000    0.0001
##    360        0.5623             nan     0.1000   -0.0004
##    380        0.5429             nan     0.1000   -0.0003
##    400        0.5239             nan     0.1000   -0.0004
##    420        0.5046             nan     0.1000    0.0000
##    440        0.4868             nan     0.1000    0.0002
##    460        0.4684             nan     0.1000   -0.0001
##    480        0.4538             nan     0.1000   -0.0001
##    500        0.4381             nan     0.1000   -0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3742             nan     0.1000    0.0051
##      2        1.3623             nan     0.1000    0.0038
##      3        1.3515             nan     0.1000    0.0048
##      4        1.3419             nan     0.1000    0.0032
##      5        1.3320             nan     0.1000    0.0042
##      6        1.3218             nan     0.1000    0.0038
##      7        1.3121             nan     0.1000    0.0041
##      8        1.3023             nan     0.1000    0.0042
##      9        1.2926             nan     0.1000    0.0031
##     10        1.2857             nan     0.1000    0.0028
##     20        1.2238             nan     0.1000    0.0015
##     40        1.1290             nan     0.1000    0.0009
##     60        1.0662             nan     0.1000    0.0004
##     80        1.0123             nan     0.1000    0.0003
##    100        0.9691             nan     0.1000   -0.0004
##    120        0.9293             nan     0.1000   -0.0002
##    140        0.8915             nan     0.1000    0.0001
##    160        0.8574             nan     0.1000   -0.0002
##    180        0.8278             nan     0.1000    0.0000
##    200        0.7977             nan     0.1000    0.0001
##    220        0.7701             nan     0.1000   -0.0001
##    240        0.7441             nan     0.1000   -0.0002
##    260        0.7187             nan     0.1000   -0.0001
##    280        0.6938             nan     0.1000   -0.0005
##    300        0.6721             nan     0.1000   -0.0001
##    320        0.6509             nan     0.1000   -0.0006
##    340        0.6312             nan     0.1000   -0.0001
##    360        0.6119             nan     0.1000   -0.0003
##    380        0.5951             nan     0.1000   -0.0001
##    400        0.5767             nan     0.1000   -0.0003
```

```
##    420        0.5599             nan     0.1000   -0.0004
##    440        0.5434             nan     0.1000   -0.0001
##    460        0.5269             nan     0.1000   -0.0002
##    480        0.5110             nan     0.1000   -0.0001
##    500        0.4969             nan     0.1000   -0.0001
##
## Elapsed training time for  GBM with  500 trees and shrinkage 0.1 is  233.611  seconds
## Validation error for GBM is  0.2646667Elapsed time for Training Random Forest with 500 trees is  280.784  s
## Validation Error rate for Random Forest with 500 trees is 0.288
```

```
## [[1]]
## Stochastic Gradient Boosting
##
## 1500 samples
## 5000 predictors
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 1200, 1200, 1200, 1200, 1200
## Resampling results:
##
##   Accuracy   Kappa
##   0.7353333  0.4706362
##
## Tuning parameter 'n.trees' was held constant at a value of 500
##  1
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
##
##
## [[2]]
##
## Call:
##  randomForest(x = image_features, y = as.factor(image_labels),      ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 70
##
##         OOB estimate of  error rate: 28.8%
## Confusion matrix:
##     0   1 class.error
## 0 517 227   0.3051075
## 1 205 551   0.2711640
```

```
train_models(modified_data_train, labels_train, full_feature = FALSE, run_cv = run.cv, run_OOB = run.OOB, K = 
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2690             nan     0.1000    0.0558
##      2        1.1731             nan     0.1000    0.0468
##      3        1.0849             nan     0.1000    0.0440
##      4        1.0123             nan     0.1000    0.0356
##      5        0.9454             nan     0.1000    0.0326
##      6        0.8857             nan     0.1000    0.0285
##      7        0.8340             nan     0.1000    0.0256
##      8        0.7860             nan     0.1000    0.0229
##      9        0.7415             nan     0.1000    0.0207
##     10        0.7026             nan     0.1000    0.0190
##     20        0.4466             nan     0.1000    0.0077
##     40        0.2285             nan     0.1000    0.0028
```

```
##     60        0.1343             nan     0.1000    0.0013
##     80        0.0833             nan     0.1000    0.0010
##    100        0.0580             nan     0.1000    0.0005
##    120        0.0435             nan     0.1000   -0.0000
##    140        0.0328             nan     0.1000    0.0003
##    160        0.0249             nan     0.1000    0.0002
##    180        0.0203             nan     0.1000    0.0000
##    200        0.0159             nan     0.1000    0.0002
##    220        0.0127             nan     0.1000    0.0000
##    240        0.0102             nan     0.1000    0.0000
##    260        0.0087             nan     0.1000   -0.0000
##    280        0.0068             nan     0.1000    0.0000
##    300        0.0056             nan     0.1000   -0.0000
##    320        0.0045             nan     0.1000   -0.0000
##    340        0.0034             nan     0.1000   -0.0000
##    360        0.0029             nan     0.1000   -0.0000
##    380        0.0024             nan     0.1000    0.0000
##    400        0.0020             nan     0.1000   -0.0000
##    420        0.0016             nan     0.1000    0.0000
##    440        0.0013             nan     0.1000    0.0000
##    460        0.0011             nan     0.1000   -0.0000
##    480        0.0009             nan     0.1000    0.0000
##    500        0.0007             nan     0.1000   -0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2803             nan     0.1000    0.0495
##      2        1.1814             nan     0.1000    0.0488
##      3        1.0975             nan     0.1000    0.0415
##      4        1.0240             nan     0.1000    0.0376
##      5        0.9609             nan     0.1000    0.0310
##      6        0.8997             nan     0.1000    0.0300
##      7        0.8509             nan     0.1000    0.0228
##      8        0.8013             nan     0.1000    0.0245
##      9        0.7573             nan     0.1000    0.0212
##     10        0.7187             nan     0.1000    0.0179
##     20        0.4566             nan     0.1000    0.0088
##     40        0.2270             nan     0.1000    0.0037
##     60        0.1303             nan     0.1000    0.0014
##     80        0.0790             nan     0.1000    0.0010
##    100        0.0539             nan     0.1000    0.0005
##    120        0.0420             nan     0.1000   -0.0000
##    140        0.0303             nan     0.1000    0.0003
##    160        0.0248             nan     0.1000    0.0001
##    180        0.0188             nan     0.1000   -0.0000
##    200        0.0153             nan     0.1000    0.0001
##    220        0.0116             nan     0.1000    0.0001
##    240        0.0094             nan     0.1000    0.0001
##    260        0.0079             nan     0.1000   -0.0000
##    280        0.0064             nan     0.1000   -0.0000
##    300        0.0048             nan     0.1000    0.0000
##    320        0.0040             nan     0.1000    0.0000
##    340        0.0032             nan     0.1000    0.0000
##    360        0.0027             nan     0.1000    0.0000
##    380        0.0022             nan     0.1000    0.0000
##    400        0.0018             nan     0.1000    0.0000
##    420        0.0015             nan     0.1000    0.0000
##    440        0.0012             nan     0.1000   -0.0000
##    460        0.0010             nan     0.1000    0.0000
##    480        0.0008             nan     0.1000   -0.0000
##    500        0.0006             nan     0.1000    0.0000
```

```
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2742             nan     0.1000    0.0565
##      2        1.1804             nan     0.1000    0.0460
##      3        1.0919             nan     0.1000    0.0417
##      4        1.0188             nan     0.1000    0.0360
##      5        0.9530             nan     0.1000    0.0327
##      6        0.8957             nan     0.1000    0.0272
##      7        0.8421             nan     0.1000    0.0260
##      8        0.7946             nan     0.1000    0.0222
##      9        0.7514             nan     0.1000    0.0207
##     10        0.7121             nan     0.1000    0.0182
##     20        0.4510             nan     0.1000    0.0087
##     40        0.2233             nan     0.1000    0.0031
##     60        0.1267             nan     0.1000    0.0017
##     80        0.0800             nan     0.1000    0.0005
##    100        0.0542             nan     0.1000    0.0005
##    120        0.0399             nan     0.1000    0.0003
##    140        0.0296             nan     0.1000    0.0002
##    160        0.0230             nan     0.1000    0.0002
##    180        0.0181             nan     0.1000    0.0001
##    200        0.0144             nan     0.1000    0.0001
##    220        0.0115             nan     0.1000   -0.0000
##    240        0.0091             nan     0.1000   -0.0000
##    260        0.0071             nan     0.1000    0.0000
##    280        0.0055             nan     0.1000    0.0000
##    300        0.0047             nan     0.1000    0.0000
##    320        0.0037             nan     0.1000    0.0000
##    340        0.0031             nan     0.1000   -0.0000
##    360        0.0023             nan     0.1000    0.0000
##    380        0.0019             nan     0.1000    0.0000
##    400        0.0015             nan     0.1000   -0.0000
##    420        0.0012             nan     0.1000    0.0000
##    440        0.0009             nan     0.1000    0.0000
##    460        0.0008             nan     0.1000    0.0000
##    480        0.0006             nan     0.1000    0.0000
##    500        0.0005             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2747             nan     0.1000    0.0540
##      2        1.1796             nan     0.1000    0.0463
##      3        1.0926             nan     0.1000    0.0439
##      4        1.0204             nan     0.1000    0.0350
##      5        0.9502             nan     0.1000    0.0337
##      6        0.8902             nan     0.1000    0.0285
##      7        0.8398             nan     0.1000    0.0252
##      8        0.7947             nan     0.1000    0.0220
##      9        0.7537             nan     0.1000    0.0182
##     10        0.7123             nan     0.1000    0.0198
##     20        0.4555             nan     0.1000    0.0080
##     40        0.2354             nan     0.1000    0.0026
##     60        0.1382             nan     0.1000    0.0016
##     80        0.0866             nan     0.1000    0.0005
##    100        0.0612             nan     0.1000   -0.0000
##    120        0.0436             nan     0.1000    0.0003
##    140        0.0331             nan     0.1000    0.0002
##    160        0.0267             nan     0.1000    0.0001
##    180        0.0203             nan     0.1000    0.0000
##    200        0.0177             nan     0.1000    0.0001
##    220        0.0140             nan     0.1000    0.0001
```

```
##    240        0.0108             nan     0.1000    0.0000
##    260        0.0091             nan     0.1000   -0.0000
##    280        0.0073             nan     0.1000    0.0000
##    300        0.0064             nan     0.1000    0.0000
##    320        0.0048             nan     0.1000    0.0000
##    340        0.0041             nan     0.1000    0.0000
##    360        0.0034             nan     0.1000    0.0000
##    380        0.0028             nan     0.1000    0.0000
##    400        0.0024             nan     0.1000   -0.0000
##    420        0.0019             nan     0.1000    0.0000
##    440        0.0016             nan     0.1000   -0.0000
##    460        0.0013             nan     0.1000    0.0000
##    480        0.0010             nan     0.1000    0.0000
##    500        0.0009             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2749             nan     0.1000    0.0567
##      2        1.1805             nan     0.1000    0.0460
##      3        1.0927             nan     0.1000    0.0430
##      4        1.0181             nan     0.1000    0.0359
##      5        0.9555             nan     0.1000    0.0292
##      6        0.8955             nan     0.1000    0.0301
##      7        0.8400             nan     0.1000    0.0274
##      8        0.7946             nan     0.1000    0.0213
##      9        0.7494             nan     0.1000    0.0213
##     10        0.7109             nan     0.1000    0.0180
##     20        0.4448             nan     0.1000    0.0087
##     40        0.2261             nan     0.1000    0.0026
##     60        0.1286             nan     0.1000    0.0012
##     80        0.0817             nan     0.1000    0.0007
##    100        0.0574             nan     0.1000    0.0001
##    120        0.0403             nan     0.1000    0.0004
##    140        0.0300             nan     0.1000    0.0002
##    160        0.0229             nan     0.1000    0.0001
##    180        0.0181             nan     0.1000    0.0000
##    200        0.0141             nan     0.1000    0.0001
##    220        0.0110             nan     0.1000    0.0001
##    240        0.0087             nan     0.1000    0.0001
##    260        0.0069             nan     0.1000    0.0000
##    280        0.0054             nan     0.1000    0.0000
##    300        0.0043             nan     0.1000   -0.0000
##    320        0.0032             nan     0.1000    0.0000
##    340        0.0026             nan     0.1000    0.0000
##    360        0.0021             nan     0.1000   -0.0000
##    380        0.0018             nan     0.1000   -0.0000
##    400        0.0013             nan     0.1000    0.0000
##    420        0.0011             nan     0.1000    0.0000
##    440        0.0009             nan     0.1000    0.0000
##    460        0.0007             nan     0.1000    0.0000
##    480        0.0006             nan     0.1000    0.0000
##    500        0.0005             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2742             nan     0.1000    0.0561
##      2        1.1792             nan     0.1000    0.0464
##      3        1.0962             nan     0.1000    0.0381
##      4        1.0183             nan     0.1000    0.0389
##      5        0.9523             nan     0.1000    0.0321
##      6        0.8929             nan     0.1000    0.0275
##      7        0.8393             nan     0.1000    0.0265
```

```
##      8        0.7930             nan     0.1000    0.0221
##      9        0.7512             nan     0.1000    0.0197
##     10        0.7106             nan     0.1000    0.0199
##     20        0.4515             nan     0.1000    0.0104
##     40        0.2316             nan     0.1000    0.0022
##     60        0.1339             nan     0.1000    0.0015
##     80        0.0848             nan     0.1000    0.0008
##    100        0.0555             nan     0.1000    0.0006
##    120        0.0426             nan     0.1000    0.0002
##    140        0.0336             nan     0.1000    0.0001
##    160        0.0266             nan     0.1000   -0.0000
##    180        0.0211             nan     0.1000    0.0000
##    200        0.0172             nan     0.1000   -0.0000
##    220        0.0140             nan     0.1000   -0.0000
##    240        0.0109             nan     0.1000    0.0000
##    260        0.0088             nan     0.1000    0.0000
##    280        0.0075             nan     0.1000   -0.0000
##    300        0.0062             nan     0.1000   -0.0000
##    320        0.0051             nan     0.1000    0.0000
##    340        0.0040             nan     0.1000    0.0000
##    360        0.0032             nan     0.1000    0.0000
##    380        0.0025             nan     0.1000    0.0000
##    400        0.0021             nan     0.1000   -0.0000
##    420        0.0018             nan     0.1000    0.0000
##    440        0.0015             nan     0.1000   -0.0000
##    460        0.0013             nan     0.1000   -0.0000
##    480        0.0010             nan     0.1000    0.0000
##    500        0.0008             nan     0.1000   -0.0000
##
## Elapsed training time for  GBM with  500 trees and shrinkage 0.1 is  102.893  seconds
## Validation error for GBM is  0.0039978Elapsed time for Training Random Forest with 500 trees is  54.141  se
## Validation Error rate for Random Forest with 500 trees is 0.004

## [[1]]
## Stochastic Gradient Boosting
##
## 1500 samples
## 2131 predictors
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 1200, 1201, 1200, 1200, 1199
## Resampling results:
##
##   Accuracy   Kappa
##   0.9960022  0.9920032
##
## Tuning parameter 'n.trees' was held constant at a value of 500
##  1
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
##
##
## [[2]]
##
## Call:
##  randomForest(x = image_features, y = as.factor(image_labels),      ntree = 500)
##                Type of random forest: classification
```

```
##                      Number of trees: 500
## No. of variables tried at each split: 46
##
##         OOB estimate of  error rate: 0.4%
## Confusion matrix:
##     0   1 class.error
## 0 739   5 0.006720430
## 1   1 755 0.001322751
```

GBM Cross Validation Results

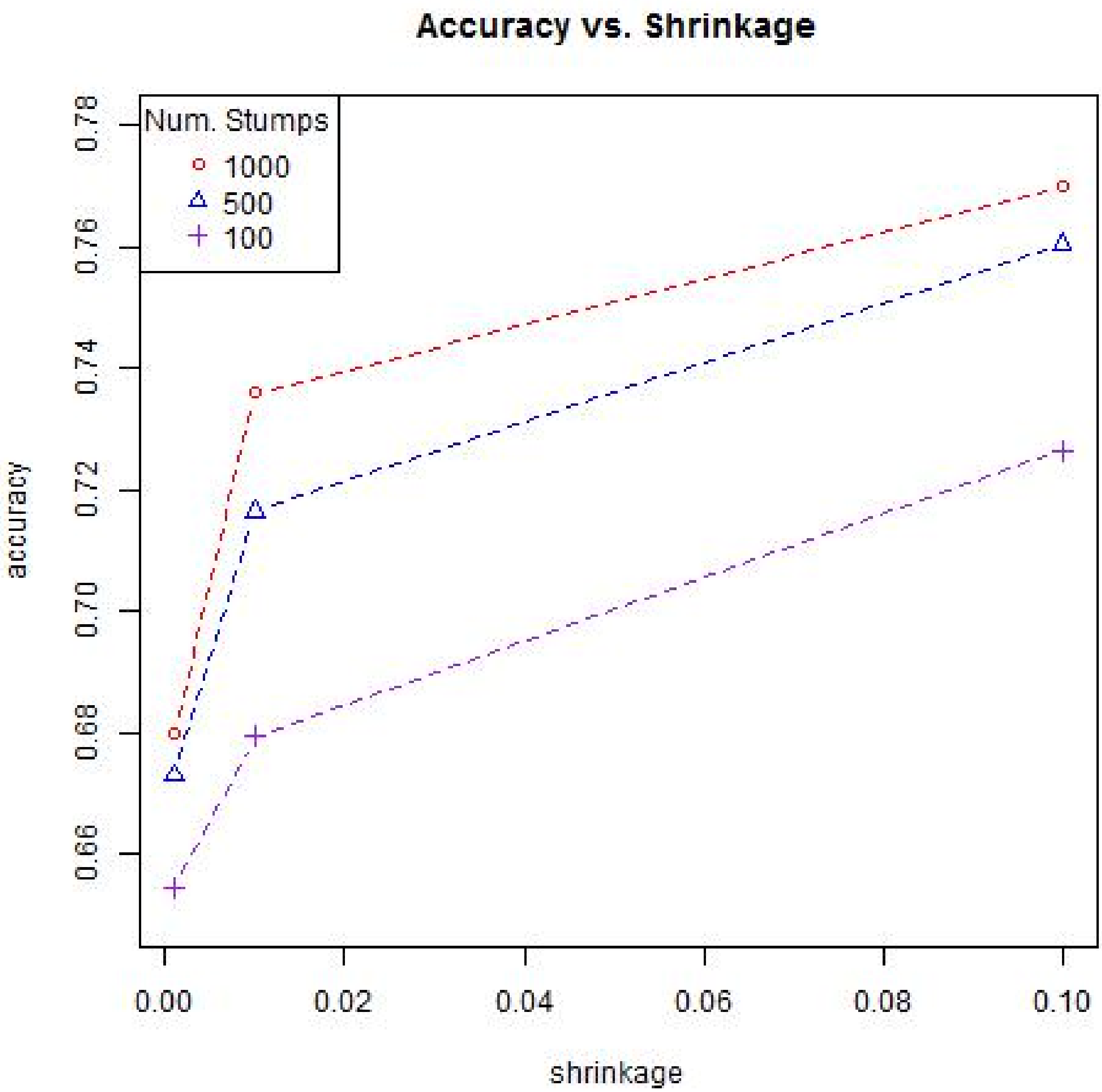

Figure 1: Figure 1: GBM Cross Validation Results

As can be seen in the above figure, a shrinkage value of 0.1 appears to be the best choice regardless of the number of trees. At a shrinkage value of 0.1, the 500 tree and 1000 tree model have nearly identical errors.

What is the best choice of parameters? Though the 1000 tree model is slightly better than the 500 tree model when shrinkage is 0.1, the 500 tree model is chosen to avoid overfitting. Additionally, the 500 tree model trains quicker, predicts quicker, and is smaller to store, so given the scenario of creating a phone app, these considerations make the 500 tree model more appropriate.

Random Forest OOB Results

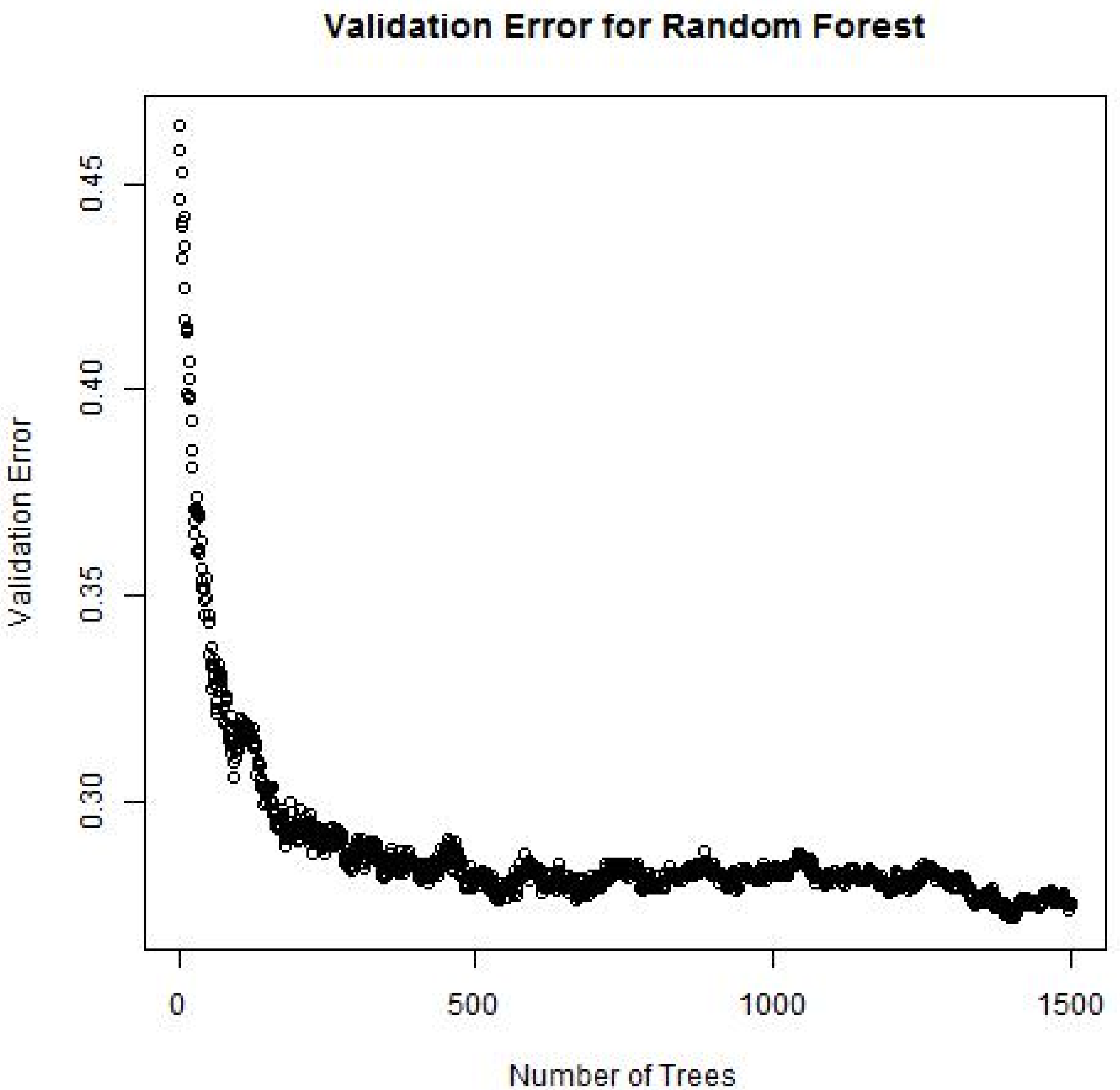

Figure 2: Figure 2: Random Forest OOB error results

As expected, the above results show that, as the number of trees increases, the OOB error decreases at a very high rate until it eventually flat lines.

Choose the best number of trees The best number of trees to chose is the least complex model that achieves the best error. The diagram above shows that the error from 500 onwards is fairly flat, and thus we chose to use a 500 tree model for our random forest.

Step 5: Make predictions on test data

For original features

Predictions are made by the GBM model and Random Forest model on the original SIFT feature set. These predictions are on the test set, which contain 25% of the original data (i.e. 500 points).

```
tm_test=NA
if(run.test){

  load(gbm_model_original_features)
  load(rf_model_original_features)
  test_models(tune_gbm, image_rf, original_data_test, full_feature = TRUE)
  rf_predict = read.csv(rf_model_original_predict)$x
  gbm_predict = read.csv(gbm_model_original_predict)$x
  test_labels = unlist(read.csv(labels_test))
  rf_error = sum(rf_predict != test_labels)/length(test_labels)
  gbm_error = sum(gbm_predict != test_labels)/length(test_labels)
  cat("GBM error for original features is ", gbm_error, "/n")
  cat("Random Forest error for original features is, ", rf_error, "/n")

  #load(file=paste0("../output/feature_", "zip", "_", "test", ".RData"))
  #load(file="../output/fit_train.RData")
  #tm_test <- system.time(pred_test <- test(fit_train, dat_test))
  #save(pred_test, file="../output/pred_test.RData")
}
```

```
## Elapsed prediction time for  GBM with  500  trees is  1.192  seconds
## Elapsed prediction time for  Random Forest with 500 trees is  1.667  seconds
## GBM error for original features is  0.244 /nRandom Forest error for original features is,  0.304 /n
```

For test feature

Predictions are made by the GBM model and Random Forest model on the modified data set, which contains the small subset of SIFT features and additional grayscale features. These predictions are on the test set, which contain 25% of the original data (i.e. 500 points).

```
tm_test=NA
if(run.test){
  load(gbm_model_modified_features)
  load(rf_model_modified_features)
  test_models(tune_gbm, image_rf, modified_data_test, full_feature = FALSE)
  rf_predict = read.csv(rf_model_modified_predict)$x
  gbm_predict = read.csv(gbm_model_modified_predict)$x
  test_labels = unlist(read.csv(labels_test))
  rf_error = sum(rf_predict != test_labels)/length(test_labels)
  gbm_error = sum(gbm_predict != test_labels)/length(test_labels)
  cat("GBM error for modified features is ", gbm_error, "/n")
  cat("Random Forest error for modified features is, ", rf_error, "/n")

  #load(file=paste0("../output/feature_", "zip", "_", "test", ".RData"))
  #load(file="../output/fit_train.RData")
  #tm_test <- system.time(pred_test <- test(fit_train, dat_test))
  #save(pred_test, file="../output/pred_test.RData")
}
```

```
## Elapsed prediction time for  GBM with  500  trees is  0.903  seconds
## Elapsed prediction time for  Random Forest with 500 trees is  0.89  seconds
## GBM error for modified features is  0.01 /nRandom Forest error for modified features is,  0.008 /n
```

Summarize Performance of various models

While prediction performance matters, so does the running times for constructing features and testing model, given the scenario limiations of the phone app. We assume training time is not an important factor as training can be done offline on a powerful machine.

		Full SIFT Train	Full SIFT Test	Small SIFT Train	Small SIFT Test	Small SIFT+Grayscale Train	Small SIFT+Grayscale Test
GBM	Error	0.2519	0.238	0.2426	0.24	0.0033333	0.006
	Time	282 sec	1.22 sec	113 sec	0.52 sec	134 sec	0.86 seconds
	Size	19.09 MB		15.06MB		15.9MB	
Random Forest	Error	0.288	0.286	0.27	0.26	0.004666	0.006
	Time	356 sec	2.45 sec	165 sec	0.93 sec	75.87 sec	0.72 sec
	Size	2.227MB		1.663MB		.17MB	
SVM Linear	Error	0.89	0.51	0.288	0.2	0.19	0.132
	Time	134.34sec	15.12sec	67.17 sec	7.56 sec	67.94 sec	7.97 sec
	Size						

Figure 3: Figure 3: Running Time, Error, and Storage Space of Various Models

The figure above shows the results from training and testing three different feature combinations: 1) The original SIFT data 2) The smaller subset of SIFT data 3) The smaller subset of SIFT Data combined with grayscale data. First focus on the error portion of the gray collumn, which represents training error. One will notice that adding grayscale feature significantly reduced error from ~20% to ~1%. This means, despite removing RGB features, color was still a very important indicator to distinguish between poodles and fried chicken. It is noteable that Linear SVM performs significantly worse than GBM and Random Forest on the third set of features. It is also important to look at the storage size of the blue collumns. This indicates the size required to store the trained model. One will notice than Random Forest takes significantly less space to store than GBM. As such, we chose Random Forest on the third feature set as our model due to its combination of accuracy, small storage size, and quick predicting time.