# ADS Group 12 - Project 3

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")
# 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  684.006  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
```

```
## Loading required package: plyr
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3733             nan     0.1000    0.0054
##      2        1.3618             nan     0.1000    0.0043
##      3        1.3512             nan     0.1000    0.0039
##      4        1.3394             nan     0.1000    0.0041
##      5        1.3294             nan     0.1000    0.0037
##      6        1.3201             nan     0.1000    0.0032
##      7        1.3107             nan     0.1000    0.0037
##      8        1.3021             nan     0.1000    0.0026
##      9        1.2933             nan     0.1000    0.0033
##     10        1.2857             nan     0.1000    0.0017
##     20        1.2131             nan     0.1000    0.0017
##     40        1.1158             nan     0.1000    0.0002
##     60        1.0427             nan     0.1000    0.0007
##     80        0.9841             nan     0.1000    0.0002
##    100        0.9354             nan     0.1000   -0.0003
```

```
##    120        0.8914             nan     0.1000    0.0001
##    140        0.8508             nan     0.1000   -0.0000
##    160        0.8140             nan     0.1000   -0.0004
##    180        0.7796             nan     0.1000    0.0003
##    200        0.7479             nan     0.1000   -0.0006
##    220        0.7201             nan     0.1000   -0.0006
##    240        0.6901             nan     0.1000   -0.0007
##    260        0.6608             nan     0.1000   -0.0002
##    280        0.6352             nan     0.1000   -0.0002
##    300        0.6106             nan     0.1000   -0.0008
##    320        0.5873             nan     0.1000   -0.0001
##    340        0.5663             nan     0.1000   -0.0002
##    360        0.5445             nan     0.1000   -0.0009
##    380        0.5240             nan     0.1000   -0.0001
##    400        0.5050             nan     0.1000   -0.0002
##    420        0.4868             nan     0.1000   -0.0001
##    440        0.4705             nan     0.1000    0.0000
##    460        0.4543             nan     0.1000   -0.0003
##    480        0.4373             nan     0.1000   -0.0001
##    500        0.4231             nan     0.1000   -0.0004
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3765             nan     0.1000    0.0018
##      2        1.3658             nan     0.1000    0.0024
##      3        1.3544             nan     0.1000    0.0041
##      4        1.3453             nan     0.1000    0.0025
##      5        1.3360             nan     0.1000    0.0038
##      6        1.3258             nan     0.1000    0.0042
##      7        1.3174             nan     0.1000    0.0029
##      8        1.3110             nan     0.1000    0.0014
##      9        1.3025             nan     0.1000    0.0022
##     10        1.2935             nan     0.1000    0.0035
##     20        1.2208             nan     0.1000    0.0012
##     40        1.1214             nan     0.1000    0.0018
##     60        1.0463             nan     0.1000    0.0008
##     80        0.9849             nan     0.1000    0.0008
##    100        0.9349             nan     0.1000   -0.0002
##    120        0.8888             nan     0.1000   -0.0007
##    140        0.8484             nan     0.1000    0.0001
##    160        0.8095             nan     0.1000    0.0001
##    180        0.7755             nan     0.1000    0.0002
##    200        0.7401             nan     0.1000   -0.0001
##    220        0.7104             nan     0.1000    0.0001
##    240        0.6792             nan     0.1000    0.0003
##    260        0.6540             nan     0.1000   -0.0005
##    280        0.6293             nan     0.1000   -0.0004
##    300        0.6061             nan     0.1000   -0.0003
##    320        0.5852             nan     0.1000   -0.0001
##    340        0.5624             nan     0.1000   -0.0004
##    360        0.5423             nan     0.1000   -0.0003
##    380        0.5221             nan     0.1000   -0.0001
##    400        0.5035             nan     0.1000   -0.0000
##    420        0.4862             nan     0.1000   -0.0001
##    440        0.4684             nan     0.1000   -0.0002
```

```
##    460        0.4515             nan     0.1000    0.0000
##    480        0.4345             nan     0.1000   -0.0001
##    500        0.4199             nan     0.1000   -0.0005
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3736             nan     0.1000    0.0046
##      2        1.3612             nan     0.1000    0.0051
##      3        1.3509             nan     0.1000    0.0046
##      4        1.3414             nan     0.1000    0.0031
##      5        1.3293             nan     0.1000    0.0036
##      6        1.3211             nan     0.1000    0.0031
##      7        1.3117             nan     0.1000    0.0022
##      8        1.3037             nan     0.1000    0.0024
##      9        1.2957             nan     0.1000    0.0026
##     10        1.2875             nan     0.1000    0.0027
##     20        1.2149             nan     0.1000    0.0026
##     40        1.1148             nan     0.1000    0.0008
##     60        1.0428             nan     0.1000    0.0003
##     80        0.9830             nan     0.1000   -0.0001
##    100        0.9303             nan     0.1000   -0.0002
##    120        0.8838             nan     0.1000    0.0004
##    140        0.8439             nan     0.1000    0.0001
##    160        0.8053             nan     0.1000   -0.0001
##    180        0.7713             nan     0.1000   -0.0006
##    200        0.7407             nan     0.1000   -0.0001
##    220        0.7102             nan     0.1000    0.0003
##    240        0.6836             nan     0.1000   -0.0004
##    260        0.6569             nan     0.1000   -0.0003
##    280        0.6295             nan     0.1000   -0.0002
##    300        0.6055             nan     0.1000   -0.0004
##    320        0.5827             nan     0.1000   -0.0005
##    340        0.5626             nan     0.1000   -0.0004
##    360        0.5434             nan     0.1000   -0.0001
##    380        0.5218             nan     0.1000   -0.0003
##    400        0.5034             nan     0.1000   -0.0005
##    420        0.4839             nan     0.1000   -0.0002
##    440        0.4674             nan     0.1000    0.0001
##    460        0.4498             nan     0.1000   -0.0002
##    480        0.4348             nan     0.1000   -0.0000
##    500        0.4196             nan     0.1000   -0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3753             nan     0.1000    0.0040
##      2        1.3631             nan     0.1000    0.0037
##      3        1.3511             nan     0.1000    0.0052
##      4        1.3420             nan     0.1000    0.0026
##      5        1.3339             nan     0.1000    0.0020
##      6        1.3246             nan     0.1000    0.0034
##      7        1.3139             nan     0.1000    0.0045
##      8        1.3060             nan     0.1000    0.0026
##      9        1.2977             nan     0.1000    0.0014
##     10        1.2899             nan     0.1000    0.0024
##     20        1.2242             nan     0.1000    0.0017
##     40        1.1288             nan     0.1000    0.0019
```

```
##     60        1.0571             nan     0.1000    0.0016
##     80        1.0006             nan     0.1000    0.0001
##    100        0.9499             nan     0.1000    0.0004
##    120        0.9025             nan     0.1000   -0.0003
##    140        0.8622             nan     0.1000   -0.0001
##    160        0.8270             nan     0.1000   -0.0005
##    180        0.7939             nan     0.1000    0.0003
##    200        0.7613             nan     0.1000   -0.0001
##    220        0.7306             nan     0.1000    0.0001
##    240        0.7013             nan     0.1000   -0.0004
##    260        0.6729             nan     0.1000   -0.0000
##    280        0.6491             nan     0.1000   -0.0008
##    300        0.6258             nan     0.1000   -0.0005
##    320        0.6008             nan     0.1000   -0.0004
##    340        0.5790             nan     0.1000   -0.0001
##    360        0.5572             nan     0.1000    0.0002
##    380        0.5379             nan     0.1000   -0.0002
##    400        0.5205             nan     0.1000   -0.0003
##    420        0.5043             nan     0.1000   -0.0001
##    440        0.4896             nan     0.1000   -0.0004
##    460        0.4717             nan     0.1000   -0.0001
##    480        0.4558             nan     0.1000   -0.0003
##    500        0.4406             nan     0.1000   -0.0001
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3747             nan     0.1000    0.0046
##      2        1.3644             nan     0.1000    0.0042
##      3        1.3542             nan     0.1000    0.0038
##      4        1.3441             nan     0.1000    0.0033
##      5        1.3358             nan     0.1000    0.0024
##      6        1.3249             nan     0.1000    0.0040
##      7        1.3155             nan     0.1000    0.0029
##      8        1.3073             nan     0.1000    0.0035
##      9        1.2995             nan     0.1000    0.0032
##     10        1.2911             nan     0.1000    0.0016
##     20        1.2327             nan     0.1000   -0.0001
##     40        1.1340             nan     0.1000    0.0008
##     60        1.0651             nan     0.1000    0.0006
##     80        1.0076             nan     0.1000   -0.0006
##    100        0.9571             nan     0.1000    0.0002
##    120        0.9145             nan     0.1000   -0.0006
##    140        0.8730             nan     0.1000    0.0003
##    160        0.8338             nan     0.1000    0.0002
##    180        0.7979             nan     0.1000   -0.0000
##    200        0.7649             nan     0.1000   -0.0007
##    220        0.7324             nan     0.1000    0.0000
##    240        0.7009             nan     0.1000   -0.0000
##    260        0.6731             nan     0.1000   -0.0003
##    280        0.6473             nan     0.1000   -0.0007
##    300        0.6219             nan     0.1000   -0.0001
##    320        0.5988             nan     0.1000   -0.0002
##    340        0.5771             nan     0.1000   -0.0006
##    360        0.5577             nan     0.1000   -0.0003
##    380        0.5370             nan     0.1000    0.0000
```

```
##    400        0.5185             nan     0.1000   -0.0000
##    420        0.5007             nan     0.1000   -0.0000
##    440        0.4844             nan     0.1000    0.0003
##    460        0.4690             nan     0.1000   -0.0001
##    480        0.4529             nan     0.1000   -0.0001
##    500        0.4383             nan     0.1000   -0.0006
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3738             nan     0.1000    0.0056
##      2        1.3644             nan     0.1000    0.0031
##      3        1.3536             nan     0.1000    0.0038
##      4        1.3438             nan     0.1000    0.0033
##      5        1.3351             nan     0.1000    0.0032
##      6        1.3273             nan     0.1000    0.0027
##      7        1.3178             nan     0.1000    0.0037
##      8        1.3083             nan     0.1000    0.0037
##      9        1.3017             nan     0.1000    0.0018
##     10        1.2917             nan     0.1000    0.0028
##     20        1.2263             nan     0.1000    0.0012
##     40        1.1336             nan     0.1000    0.0011
##     60        1.0683             nan     0.1000    0.0010
##     80        1.0108             nan     0.1000    0.0005
##    100        0.9681             nan     0.1000    0.0000
##    120        0.9269             nan     0.1000    0.0001
##    140        0.8915             nan     0.1000   -0.0001
##    160        0.8580             nan     0.1000   -0.0002
##    180        0.8270             nan     0.1000   -0.0004
##    200        0.7984             nan     0.1000   -0.0007
##    220        0.7696             nan     0.1000    0.0003
##    240        0.7422             nan     0.1000    0.0000
##    260        0.7161             nan     0.1000   -0.0004
##    280        0.6923             nan     0.1000   -0.0001
##    300        0.6686             nan     0.1000   -0.0000
##    320        0.6467             nan     0.1000   -0.0003
##    340        0.6256             nan     0.1000   -0.0001
##    360        0.6076             nan     0.1000   -0.0003
##    380        0.5914             nan     0.1000   -0.0002
##    400        0.5738             nan     0.1000   -0.0003
##    420        0.5556             nan     0.1000   -0.0000
##    440        0.5391             nan     0.1000    0.0001
##    460        0.5221             nan     0.1000   -0.0000
##    480        0.5056             nan     0.1000   -0.0004
##    500        0.4917             nan     0.1000   -0.0003
##
## Elapsed training time for  GBM with  500 trees and shrinkage 0.1 is  216.149  seconds
## Validation error for GBM is  0.2186543Elapsed time for Training Random Forest with 500 trees is  256.671  s
## Validation Error rate for Random Forest with 500 trees is 0.2786667
## [[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: 1201, 1199, 1200, 1200, 1200
## Resampling results:
##
##   Accuracy   Kappa
##   0.7813457  0.562677
##
## 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: 27.87%
## Confusion matrix:
##     0   1 class.error
## 0 534 217   0.2889481
## 1 201 548   0.2683578
```

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

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2688             nan     0.1000    0.0576
##      2        1.1715             nan     0.1000    0.0470
##      3        1.0876             nan     0.1000    0.0391
##      4        1.0095             nan     0.1000    0.0372
##      5        0.9465             nan     0.1000    0.0300
##      6        0.8866             nan     0.1000    0.0286
##      7        0.8335             nan     0.1000    0.0251
##      8        0.7871             nan     0.1000    0.0221
##      9        0.7461             nan     0.1000    0.0195
##     10        0.7081             nan     0.1000    0.0168
##     20        0.4492             nan     0.1000    0.0079
##     40        0.2333             nan     0.1000    0.0020
##     60        0.1376             nan     0.1000    0.0017
##     80        0.0909             nan     0.1000    0.0007
##    100        0.0645             nan     0.1000    0.0000
##    120        0.0483             nan     0.1000    0.0005
##    140        0.0368             nan     0.1000    0.0002
##    160        0.0288             nan     0.1000    0.0002
##    180        0.0230             nan     0.1000    0.0000
##    200        0.0184             nan     0.1000    0.0001
##    220        0.0151             nan     0.1000    0.0001
##    240        0.0124             nan     0.1000    0.0000
```

```
##    260        0.0101             nan     0.1000   -0.0000
##    280        0.0085             nan     0.1000   -0.0000
##    300        0.0068             nan     0.1000    0.0000
##    320        0.0057             nan     0.1000    0.0000
##    340        0.0046             nan     0.1000   -0.0000
##    360        0.0038             nan     0.1000   -0.0000
##    380        0.0032             nan     0.1000   -0.0000
##    400        0.0026             nan     0.1000   -0.0000
##    420        0.0021             nan     0.1000    0.0000
##    440        0.0018             nan     0.1000    0.0000
##    460        0.0015             nan     0.1000    0.0000
##    480        0.0012             nan     0.1000    0.0000
##    500        0.0010             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2675             nan     0.1000    0.0614
##      2        1.1662             nan     0.1000    0.0503
##      3        1.0828             nan     0.1000    0.0393
##      4        1.0095             nan     0.1000    0.0356
##      5        0.9419             nan     0.1000    0.0328
##      6        0.8850             nan     0.1000    0.0267
##      7        0.8327             nan     0.1000    0.0251
##      8        0.7869             nan     0.1000    0.0206
##      9        0.7411             nan     0.1000    0.0231
##     10        0.6991             nan     0.1000    0.0199
##     20        0.4483             nan     0.1000    0.0082
##     40        0.2358             nan     0.1000    0.0036
##     60        0.1382             nan     0.1000    0.0017
##     80        0.0875             nan     0.1000    0.0008
##    100        0.0637             nan     0.1000    0.0004
##    120        0.0488             nan     0.1000   -0.0001
##    140        0.0385             nan     0.1000   -0.0001
##    160        0.0307             nan     0.1000    0.0001
##    180        0.0253             nan     0.1000   -0.0000
##    200        0.0196             nan     0.1000    0.0001
##    220        0.0162             nan     0.1000    0.0001
##    240        0.0135             nan     0.1000    0.0000
##    260        0.0111             nan     0.1000    0.0001
##    280        0.0088             nan     0.1000   -0.0000
##    300        0.0072             nan     0.1000   -0.0000
##    320        0.0060             nan     0.1000   -0.0000
##    340        0.0050             nan     0.1000    0.0000
##    360        0.0041             nan     0.1000   -0.0000
##    380        0.0035             nan     0.1000   -0.0000
##    400        0.0030             nan     0.1000    0.0000
##    420        0.0024             nan     0.1000   -0.0000
##    440        0.0020             nan     0.1000   -0.0000
##    460        0.0016             nan     0.1000    0.0000
##    480        0.0014             nan     0.1000   -0.0000
##    500        0.0011             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2684             nan     0.1000    0.0574
##      2        1.1739             nan     0.1000    0.0450
```

```
##      3        1.0930             nan     0.1000    0.0369
##      4        1.0085             nan     0.1000    0.0388
##      5        0.9405             nan     0.1000    0.0333
##      6        0.8844             nan     0.1000    0.0268
##      7        0.8311             nan     0.1000    0.0263
##      8        0.7823             nan     0.1000    0.0224
##      9        0.7425             nan     0.1000    0.0188
##     10        0.7052             nan     0.1000    0.0167
##     20        0.4518             nan     0.1000    0.0078
##     40        0.2344             nan     0.1000    0.0014
##     60        0.1380             nan     0.1000    0.0018
##     80        0.0900             nan     0.1000    0.0010
##    100        0.0619             nan     0.1000    0.0007
##    120        0.0466             nan     0.1000    0.0000
##    140        0.0367             nan     0.1000    0.0000
##    160        0.0281             nan     0.1000    0.0002
##    180        0.0232             nan     0.1000    0.0002
##    200        0.0177             nan     0.1000    0.0002
##    220        0.0145             nan     0.1000    0.0001
##    240        0.0118             nan     0.1000    0.0000
##    260        0.0094             nan     0.1000    0.0001
##    280        0.0079             nan     0.1000    0.0000
##    300        0.0062             nan     0.1000    0.0000
##    320        0.0051             nan     0.1000   -0.0000
##    340        0.0043             nan     0.1000   -0.0000
##    360        0.0035             nan     0.1000   -0.0000
##    380        0.0030             nan     0.1000    0.0000
##    400        0.0024             nan     0.1000   -0.0000
##    420        0.0020             nan     0.1000   -0.0000
##    440        0.0016             nan     0.1000   -0.0000
##    460        0.0014             nan     0.1000   -0.0000
##    480        0.0011             nan     0.1000   -0.0000
##    500        0.0009             nan     0.1000   -0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2684             nan     0.1000    0.0590
##      2        1.1733             nan     0.1000    0.0478
##      3        1.0895             nan     0.1000    0.0385
##      4        1.0080             nan     0.1000    0.0395
##      5        0.9444             nan     0.1000    0.0311
##      6        0.8844             nan     0.1000    0.0299
##      7        0.8297             nan     0.1000    0.0259
##      8        0.7859             nan     0.1000    0.0209
##      9        0.7412             nan     0.1000    0.0207
##     10        0.7021             nan     0.1000    0.0178
##     20        0.4441             nan     0.1000    0.0092
##     40        0.2283             nan     0.1000    0.0033
##     60        0.1399             nan     0.1000    0.0009
##     80        0.0909             nan     0.1000    0.0011
##    100        0.0643             nan     0.1000    0.0005
##    120        0.0454             nan     0.1000    0.0004
##    140        0.0355             nan     0.1000   -0.0000
##    160        0.0283             nan     0.1000    0.0000
##    180        0.0228             nan     0.1000    0.0001
```

```
##    200        0.0184             nan     0.1000   -0.0000
##    220        0.0150             nan     0.1000    0.0000
##    240        0.0119             nan     0.1000    0.0000
##    260        0.0094             nan     0.1000    0.0000
##    280        0.0081             nan     0.1000   -0.0000
##    300        0.0065             nan     0.1000    0.0000
##    320        0.0052             nan     0.1000    0.0000
##    340        0.0043             nan     0.1000    0.0000
##    360        0.0036             nan     0.1000    0.0000
##    380        0.0029             nan     0.1000   -0.0000
##    400        0.0024             nan     0.1000    0.0000
##    420        0.0019             nan     0.1000    0.0000
##    440        0.0015             nan     0.1000    0.0000
##    460        0.0013             nan     0.1000    0.0000
##    480        0.0011             nan     0.1000   -0.0000
##    500        0.0009             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2742             nan     0.1000    0.0562
##      2        1.1717             nan     0.1000    0.0488
##      3        1.0882             nan     0.1000    0.0411
##      4        1.0151             nan     0.1000    0.0370
##      5        0.9508             nan     0.1000    0.0309
##      6        0.8937             nan     0.1000    0.0282
##      7        0.8399             nan     0.1000    0.0239
##      8        0.7902             nan     0.1000    0.0232
##      9        0.7486             nan     0.1000    0.0199
##     10        0.7129             nan     0.1000    0.0169
##     20        0.4515             nan     0.1000    0.0088
##     40        0.2385             nan     0.1000    0.0022
##     60        0.1401             nan     0.1000    0.0013
##     80        0.0908             nan     0.1000    0.0009
##    100        0.0654             nan     0.1000    0.0008
##    120        0.0520             nan     0.1000    0.0000
##    140        0.0399             nan     0.1000    0.0000
##    160        0.0333             nan     0.1000   -0.0000
##    180        0.0264             nan     0.1000   -0.0000
##    200        0.0218             nan     0.1000    0.0000
##    220        0.0186             nan     0.1000   -0.0000
##    240        0.0149             nan     0.1000    0.0001
##    260        0.0122             nan     0.1000    0.0001
##    280        0.0104             nan     0.1000   -0.0000
##    300        0.0084             nan     0.1000    0.0001
##    320        0.0069             nan     0.1000   -0.0000
##    340        0.0056             nan     0.1000    0.0000
##    360        0.0049             nan     0.1000    0.0000
##    380        0.0040             nan     0.1000   -0.0000
##    400        0.0034             nan     0.1000   -0.0000
##    420        0.0028             nan     0.1000   -0.0000
##    440        0.0025             nan     0.1000    0.0000
##    460        0.0020             nan     0.1000   -0.0000
##    480        0.0016             nan     0.1000   -0.0000
##    500        0.0013             nan     0.1000    0.0000
##
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.2677             nan     0.1000    0.0589
##      2        1.1716             nan     0.1000    0.0471
##      3        1.0897             nan     0.1000    0.0377
##      4        1.0157             nan     0.1000    0.0370
##      5        0.9494             nan     0.1000    0.0320
##      6        0.8915             nan     0.1000    0.0295
##      7        0.8382             nan     0.1000    0.0238
##      8        0.7911             nan     0.1000    0.0237
##      9        0.7496             nan     0.1000    0.0198
##     10        0.7110             nan     0.1000    0.0175
##     20        0.4538             nan     0.1000    0.0084
##     40        0.2393             nan     0.1000    0.0028
##     60        0.1426             nan     0.1000    0.0019
##     80        0.0923             nan     0.1000    0.0008
##    100        0.0634             nan     0.1000    0.0004
##    120        0.0513             nan     0.1000   -0.0000
##    140        0.0399             nan     0.1000   -0.0000
##    160        0.0332             nan     0.1000    0.0000
##    180        0.0259             nan     0.1000    0.0001
##    200        0.0204             nan     0.1000    0.0000
##    220        0.0170             nan     0.1000   -0.0000
##    240        0.0141             nan     0.1000   -0.0000
##    260        0.0123             nan     0.1000   -0.0000
##    280        0.0102             nan     0.1000   -0.0000
##    300        0.0086             nan     0.1000    0.0000
##    320        0.0071             nan     0.1000    0.0000
##    340        0.0060             nan     0.1000   -0.0000
##    360        0.0051             nan     0.1000    0.0000
##    380        0.0043             nan     0.1000   -0.0000
##    400        0.0036             nan     0.1000   -0.0000
##    420        0.0031             nan     0.1000   -0.0000
##    440        0.0026             nan     0.1000    0.0000
##    460        0.0022             nan     0.1000   -0.0000
##    480        0.0019             nan     0.1000    0.0000
##    500        0.0016             nan     0.1000   -0.0000
##
## Elapsed training time for  GBM with  500 trees and shrinkage 0.1 is  98.49  seconds
## Validation error for GBM is  0.004664452Elapsed time for Training Random Forest with 500 trees is  58.056
## Validation Error rate for Random Forest with 500 trees is 0.005333333
## [[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, 1200, 1200, 1199, 1201
## Resampling results:
##
##   Accuracy   Kappa
```

```
##   0.9953355  0.9906711
##
## 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: 0.53%
## Confusion matrix:
##     0   1 class.error
## 0 744   7 0.009320905
## 1   1 748 0.001335113
```

#### 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

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)
```

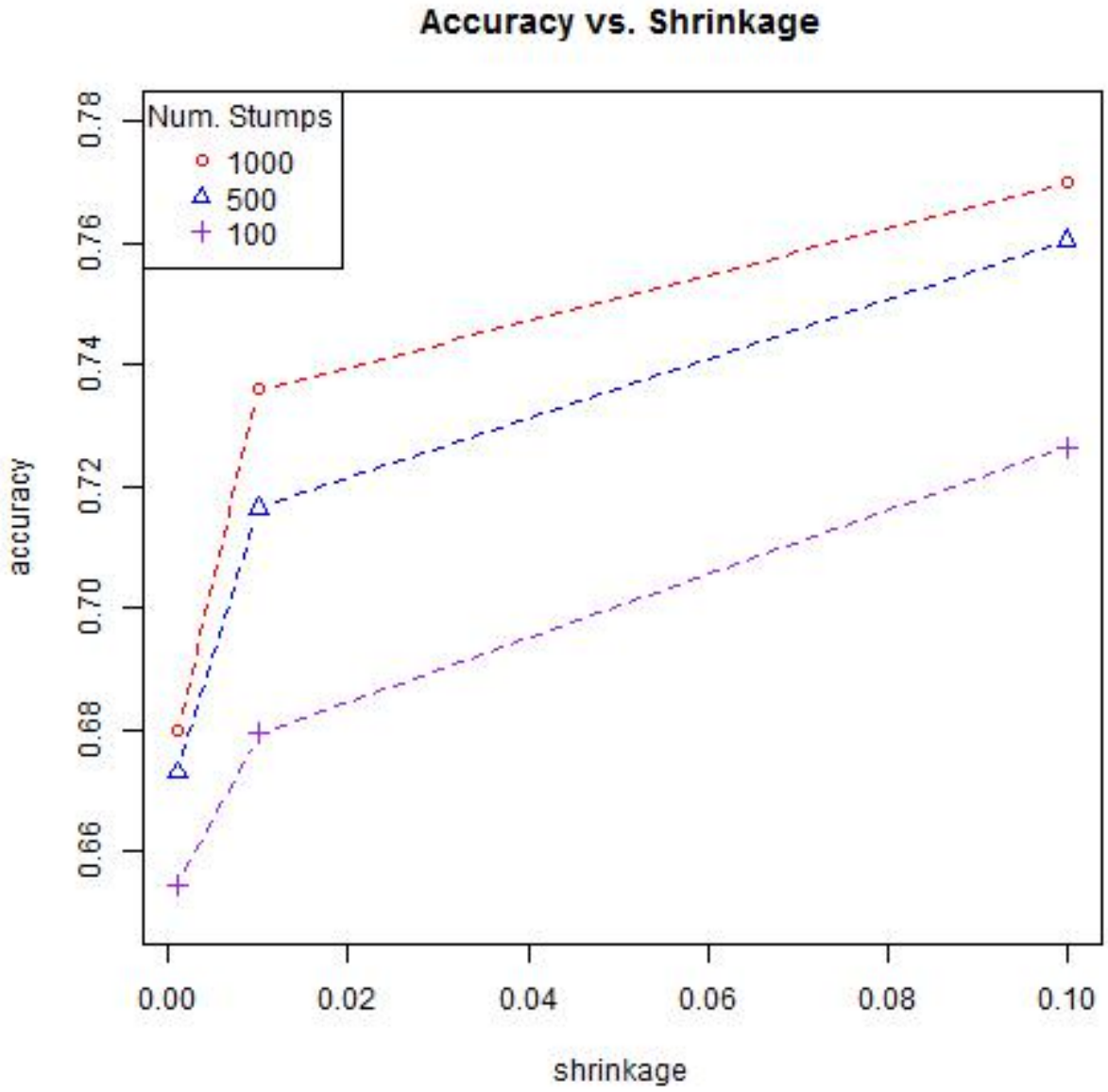


Figure 1: Figure 1: GBM Cross Validation Results

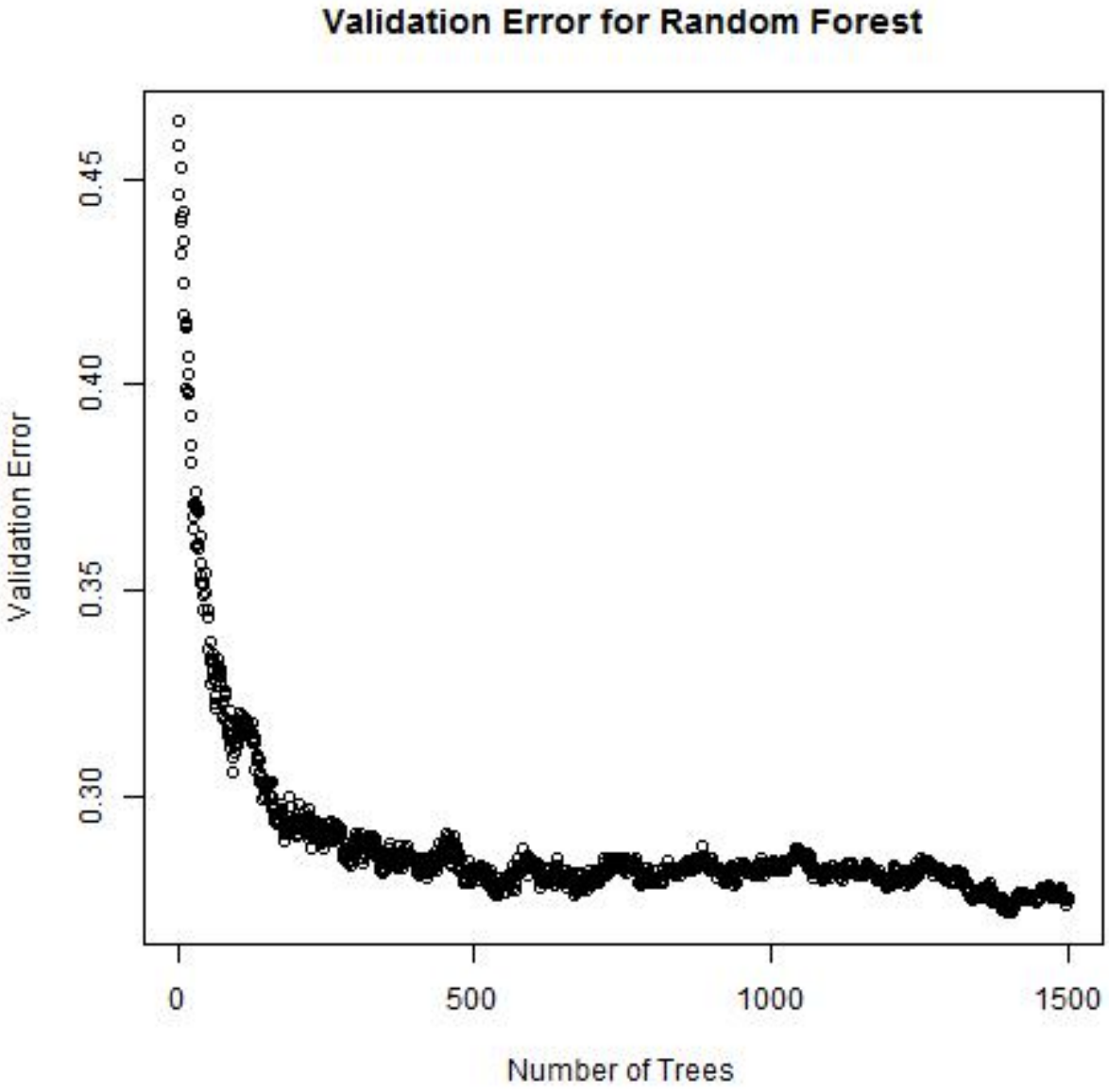


Figure 2: Figure 2: Random Forest OOB error results

```
  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  1.182  seconds
## Elapsed prediction time for  Random Forest with 500 trees is  1.435  seconds
## GBM error for original features is  0.264 /nRandom Forest error for original features is,  0.28 /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  0.734  seconds
## Elapsed prediction time for  Random Forest with 500 trees is  0.752  seconds
## GBM error for modified features is  0.004 /nRandom Forest error for modified features is,  0.002 /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.