Image Classification Based on Color Style

Final Report

Nan Ma

Shuang Su

Xiaoyu Li

03/19/2016

Santa Clara University

Github code link: https://github.com/susiesu2/ColorStyleClassification

Abstract

This study aims to classify a collection of images based on styles. In our study, we will define the style of images based on colors, gray scale, brightness, lightness and contrast. There are three stages in our study. The first stage, we extract the features with Java from the images. We also calculated the top k color clusters in each image using KMeans algorithm. Then we map each image to 64 color ranges in color space. In the second phase,we train the model using Random Forest, Support Vector Machine (SVM), Generalized Boosted Regression. In the third phase, we display the classification of images in groups in webpages. The results of this study indicates that Random Forest and Generalized Boosted Regression produces much higher correct matching percentage of the classification against human-eye pre-labeling. The highest match percentage of classification to be 48% in our tests for both the Random Forest model the the Generalized Boosted Regression model. Also the parameter k in KMeans is sensitive and could affect the output in classification. In our tests, k = 5 produces the best results for the learning models. There are also some abnormal cases, please refer to section 7 and section 8 to see more discussion and suggestions for future studies.

The following is the link to our Github code https://github.com/susiesu2/ColorStyleClassification.

Table of Contents

2. Introduction

2.1 Objective

Nowadays, many websites such as Airbnb, Redfin and Pinterest own tons of customer images. They are valuable data to generate business insights and enable recommendation system. These images contain ample implicit information which are not fully expressed in text, such as the design style, the clean level, the colorfulness of the home. If we manually extract such information by tagging the images and then do data mining based on text, the process is prone to error and requires a lot of human efforts. So we’d like to build a system to do image mining based on the major colors in an image. In our experiment, our objective is to group a huge amount of house interior images based on color style.

2.2 What is the problem

Different images contains different set of colors, which contains lots of meaningful information, i.e., telling the taste of a person from what images he/she likes. In order to tell the style of images, we want to implement a image mining model to group the images by color, that is to find images with the similar colors. Since the style of a design is highly dependent on the colors, we can roughly tell the style of an image based on its colors. If the colors in two images are similar, we say the two images have similar style. The system should output groups of images of similar style.

2.3 Why this is a project related this class

Image is one of the major data resources, which contains unbelievable hidden information. Image mining is an important component of data mining. This area is underdeveloped and has huge potential.

2.4 Scope of investigation

Image mining is a complex topic. In our research, we cut into this topic from color. In our experiment, we only look at color content of images. Other features like texture, object are not discussed in this paper. Thus our output of groups of images will only based on color information, e.g., RGB, HSV, brightness, grayscale, etc.

3. theoretical bases and literature review

3.1 Definition of the problem

This study aims to classify a collection of images based on styles. In our study, we will define the style of images based on colors, gray scale, brightness and lightness.

3.2 Theoretical Background of the Problem

Currently, images databases have a tendency to become too large to be handled manually. An extremely large number of image data such as satellite images, medical images, and digital photographs are generated every day. If these images could be analyzed, a lot of effective information can be extracted. There are mainly two major approaches for imaging mining: one is Functionally Driven Framework and the other is Information-Driven Framework [Zhang, (2001)]. Traditionally, images have been represented by alphanumeric indices for database image retrieval. However, strings and numbers are not closely related to human being's emotions and feelings. Therefore, there are new need to develop techniques for extracting features of images using other approaches, such as Content-Based Image Retrieval system [Gudivada--Raghavan (1995)].

3.3 Related research to solve the problem

Content Based Image Retrieval is a popular image retrieval system which is used to retrieve the useful features of the given image. In the research article by [Kannan, (2010)], this research article employed the CBIR (Content-Based Image Retrieval) model to retrieve the target image based on useful features of the given image, which including texture, color, shape, region and others. In the color based image retrieval system, the RGB model is used. The RGB color components are taken from each and every image. In the following step, the mean values of Red, Green, and Blue components of target images are calculated and stored in the database. Next, the study by [Kannan (2010)] performed a clustering of images using the Fuzzy C-means algorithm. Fuzzy C-means (FCM) is one of the clustering methods which allow one piece of data to belong to two or more clusters. In this clustering, each point has a degree of belonging to clusters, as in fuzzy logic, rather than belonging completely too just one cluster. Thus, points on the edge of a cluster may be in the cluster to a lesser degree than points in the center of cluster.

Another study by [Dubey (2011)] used a combination of techniques, namely Color Histogram and Edge Histogram Descriptor. The nature of the Image is based on the Human Perception of the Image. Computer vision and image processing algorithms are used to analyze the images in this research study. For color the histogram of images are computed and for edge density it is Edge Histogram Descriptor (EHD) that is found. For retrieval of images, the averages of the two techniques are made and the resultant Image is retrieved.

3.4 Advantage/disadvantage of those research

One advantage of the research article by Kannan [Kannan, (2010)] is that this study combines the concepts of CBIR and Image mining, and also develops a new clustering algorithms to increase the speed of the image retrieval system. The disadvantage of this study is that it only relies on RGB color to learn the style. It might not produce the optimal results for user query.

3.5 Our solution to solve this problem

Our approach to find similar images has two phases. The first phase is to cluster the colors in each image by k-means method, so as to find the top k major colors in an image. We also extract other color information of an image such as grayscale, brightness and lightness. The second phase is classification. First, we will train different machine learning models using a sample of images and our expected output. We use the following three models to train our model: random forests, generalized boosted regression and support vector machine. Then we use the model to do classification of the rest images.

3.6 Where our solution different from others

- We use clustering to do color quantization, i.e., group adjacent pixels into clusters, then compute a quantized color for each cluster.
- Our solution incorporates machine learning models, namely Random Forest, Support Vector Machine and Generalized boosted regression.

3.7 Why our solution is better

The style of home interior design is highly dependent on color, so color has a high importance in determining style. Color can be easily represented and summarized in number. So our approach to find similar style home using color is efficient.

4. Hypothesis

4.1 Multiple hypothesis

- We define style of an image by color, gray scale, brightness and lightness, etc.
- After the analysis, the images are grouped by style. Beforehand, we will have a look at the images and group them by human eyes. Our goal is the output of classification aligns with our expected grouping.

5. Methodology

5.1 How to generate/collect input data

The input dataset is a collection of housing interior images downloaded from Google. The size of the dataset is about 250 pictures. A random sample of the dataset will be manually classified first and then used as training data.

5.2 How to solve the problem

The general way to find similar images consist of two phases, first extract features, and then cluster the images based on extracted features. We will modify this model to meet our need to find similar style images. We define design style as a list of features like color, gray scale, brightness, and contrast. With test dataset, we classify the images into 14 different styles, and then use machine-learning models to train the test dataset. The trained model will automatically classify the rest of images into 14 styles.

5.3 Algorithm design

Our image classification algorithm includes two phases: feature extraction, data training and generating output.

5.3.1 Feature Extraction

Features like, color, gray scale, brightness, and contrast tell much about design styles. A raw image does not contain those information, thus in this step we will extract these information from pixels. The output for this phase is a labeled dataset with desired features.

There are two common color coordination system in use today, RGB(Red, Green, Blue) and HSV(Hue, Saturation, Value). We use both systems and exact as much as features we can relate to design styles. We do not decide the importance of each feature at this stage, the relevancy will be learnt by machine learning models in training stage.

Color:

The number of pixels for an image can range from 10^2 to 10^7. Each pixel contains three values: the amount of red, green, and blue. The comparison of color pixel by pixel is a burden for computation. Thus we use k-means to cluster an image into k colors, with RGB value, and weight of that color. The RGB model can represent 255^3 colors. We divide the color space into 64 pools. For each of the k clustered colors, if the color falls into the range of one pool, then we update the weight of that pool. When this is done, an image should have 64 attributes indicating the weight of each color pool in the image.

Gray scale:

Gray scale carries intensity information. The gray scale can be calculated by RGB value based on the luminance-preserving conversion equation. To represent the overall intensity of the image, we use median, first quartile third quartile value of gray scale.

Hue:

Hue is the attribute of a visual sensation according to which an area appears to be similar to one of the perceived colors: red, yellow, green, and blue, or to a combination of two of them. The calculation of hue depends on the max of RGB color channel. So there are three equations to calculate hue:

If Red is max, then Hue = (G-B)/(max-min) *60

If Green is max, then Hue = (2.0 + (B-R)/(max-min))*60

If Blue is max, then Hue = (4.0 + (R-G)/(max-min))*60

Lightness:

Under HSL model, lightness is defined as the average of RGB value. In order to represent the overall lightness of the image, we use median value of lightness.

Contrast:

Contrast is the difference of gray scale value between the largest and smallest value. We calculate it based on the k clustered colors.

5.3.2 Classification and training

5.3.2.1 Generate training data

The target vector is the style of housing. We randomly select parts of the complete dataset and manually classify the images into n (n = 14 in this study) styles. The classified dataset will be our training dataset and test dataset.

5.3.2.2 Machine learning models

Our project is supervised learning, thus we will use supervised classification models for learning purpose. In order to get the best result, we will test on several models and select the one that generates the best result.

The models we will test on are:

- Random Forest
- Generalized boosted regression
- Support Vector Machine

5.4 Programming Languages

The feature exaction will be implemented in Java. The training and machine learning will be implemented in R.

5.5 Tools used

We will use ImageIO library to get pixel data from images. Several R packages will be used for machine learning purpose, including e1071(), randomForest, and gbr (generalized boosted regression).

5.6 Output Generation

The output is a column vector representing the style of each image like the following table.

ImageName	Style
img1	DS3
img2	DS1
img3	DS3
img4	DS1
...	...
imgN	DS4

5.7 Methodology to test against hypothesis

We apply control variable method to test against both features and machine learning models. We have three models to test to see which one generates the lowest error rate. When applying one model, we change the amount of features one by one (e.g., the k parameter in the Kmeans algorithm), and try to find the best combination. We apply this comparison to all three models and finally pick the best matching one judged by test datasets. If any of the models can generate good results, then we confirm the housing style is largely determined by color content.

6 Implementation

6.1 Feature Extraction

The first phase of this study is to extract features from image collections. We implemented this phase by using the java.awt.image.BufferedImage package. The following is our code API

```
public static List<Color> readIMG(String filePath) {
    BufferedImage bi;
    List<Color> aIMG = new ArrayList<Color>();
    try {
        bi = ImageIO.read(new File(filePath));
        for (int y = 0; y < bi.getHeight(); y++) {
            for (int x = 0; x < bi.getWidth(); x++) {
                int[] pixel = bi.getRaster().getPixel(x, y, new int[3]);
                Color aColor = new Color(pixel[0], pixel[1], pixel[2]);
                aIMG.add(aColor);
            }
        }
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return aIMG;
}
```

6.2 KMeans to extraction top k colors

After we extract the pixel information (represented by red, green, blue color object), our next step is to find out the dominant top k colors by using the Kmeans algorithm to do the clusters. In this step, each images’ top k colors will be identified and saved. This step greatly reduces the dimension of colors in each picture. For instance, if a picture is about 500 by 500 pixels. Then 250000 pixels of colors are saved under the picture in the first phase of feature extraction. After we have done the Kmeans clustering, we will only store top k colors in each picture. The following are the major APIs for implementing the Kmeans algorithm.

```
public class Cluster {
	private final List<Color> points;
	private Color centroid;
	private int id;
	public Cluster(Color firstPoint) {
		points = new ArrayList<Color>();
		//addPoint(firstPoint);
		centroid = firstPoint;
	}
	public Color getCentroid() {
		return centroid;
	}
}

public class Clusters extends ArrayList<Cluster>{
	private static final long serialVersionUID = 1L;
	private final List<Color> allPoints;
	private boolean isChanged;

	public Clusters(List<Color> allPoints) {
		this.allPoints = allPoints;
	}

	/**
	 * @param point
	 * @return the index of the Cluster nearest to the point
	 */
	public Integer getNearestCluster(Color color) {
		double minSquareOfDistance = Double.MAX_VALUE;
		int itsIndex = -1;
		for (int i = 0; i < size(); i++) {
			double squareOfDistance = color.getSquareOfDistance(get(i)
					.getCentroid());
			if (squareOfDistance < minSquareOfDistance) {
				minSquareOfDistance = squareOfDistance;
				itsIndex = i;
			}
		}
		return itsIndex;
	}
}

public class KMeans {
    private static final Random random = new Random();
    public final List<Color> allPoints;
    public final int k;
    private Clusters pointClusters; // the k Clusters
    public KMeans(List<Color> allColors, int k) {
        if (k < 2)
            new Exception("The value of k should be 2 or more.")
                    .printStackTrace();
        this.k = k;
        this.allPoints = allColors;
    }
}
```

6.3 Color range map

The total number of possible colors are 255^3. In order to reduce the color range make the classification more sensitive and applicable for small dataset (e.g. a few hundreds images), we mapped all of the possible colors to 64 colorBuckets. Therefore, we will produce a table in the following format. Each image will have top five (let's take five for example) colors with a weight represented how many percentage of pixels falls into that particular color cluster. Therefore, each image have five colors mapped into the 64 colorBuckets with a weight. These parameters will be used by all the three training models to learn images' potential styles.

img ID	range 0	range 1	range 2	....	...	range 62	range 63
1	0.255	0.011	0.33	...		0.15	0.11
...	...	...	...	...	...	...	...
100	0.233	0	0.132	0	0.212	0.235	0.1515
248	0	0.334	0	0.17	0.27	0.1313	0.3124

6.4 Training Phase

In this study, we use three different training models: Random forest, Generalized boosted regression, and Support vector machine. In the beginning, we use human eye to label all the images in our dataset into 14 categories. Then we use this information and also the table in the step of color range map to train our data.

6.4.1 Random forest training model

Random forests is a notion of the general technique of random decision forests that are an ensemble learning method for classification,regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set (cited from wikipedia).

In this study, we tried to adjust the number of decision trees to fit our model. In the end, the parameter of 1000 seems a good fit for our dataset. The following is our code for random forest training model.

```
train<-read.csv("train.csv")
test<-read.csv("test.csv")
library(randomForest)
set.seed(415)
fit <- randomForest(as.factor(Style) ~ range0 + range1 + range2 + range3 + range4 + range5 +
range6 + range7 + range8 + range9 + range10 + range11 + range12 + range13 + range14 +
range15 + range16 + range17 + range18 + range19 + range20 + range21 + range22 + range23
+ range24 + range25 + range26 + range27 + range28 + range29 + range30 + range41 +
range42 + range43 + range44  + range45 + range46 + range47 + range48 + range49 + range50
+ range51 + range52 + range53 + range54 + range55 + range56 + range57 + range58 +
range59 + range60 + range61 + range62 + range63 + grayscale + contrast, data=train,
importance=TRUE, ntree=1000)
(VI_F=importance(fit))
varImpPlot(fit)
Prediction <- predict(fit, test)
submit <- data.frame(ID = test$imageID, Category = Prediction)
write.csv(submit, file = "firstforest.csv", row.names = FALSE)
```

6.4.2 Support vector machine model

Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier (cited from wikipedia). We used R to implement the support vector machine model and the following is our code.

```
library("e1071")
train<-read.csv("train.csv")
test<-read.csv("test.csv")
nb<-ncol(train)
id<-test[,1]
category<-as.character(train[,nb])
attributes<-train[,c(-1,-nb)]
testattrib<-test[,c(-1,0)]
model<-svm(attributes, category, type="C", probability = TRUE)
prediction<-predict(model, testattrib, probability = TRUE)
out<-cbind(id, as.character(prediction))
write.csv(out, file = "svm.csv", row.names = FALSE)
```

6.4.3 Generalized boosted regression model

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function (cited from wikipedia). We used R to implement this model and the following is our model.

```
train<-read.csv("train.csv")
test<-read.csv("test.csv")
library(gbm)
library(dplyr)
category = train$Style
train = select(train, -Style)
end_trn = nrow(train)
all = rbind(train, test)
end=nrow(all)
all = select(all, range0 , range1 , range2 , range3 , range4 , range5 , range6 , range7 , range8 ,
range9 , range10 , range11 , range12 , range13 , range14 , range15 , range16 , range17 ,
range18 , range19 , range20 , range21 , range22 , range23 , range24 , range25 , range26 ,
range27 , range28 , range29 , range30 , range41 , range42 , range43 , range44  , range45 ,
range46 , range47 , range48 , range49 , range50 , range51 , range52 , range53 , range54 ,
range55 , range56 , range57 , range58 , range59 , range60 , range61 , range62 , range63 ,
grayscale , contrast)
head(all)
Model = gbm.fit(x = all[1:end_trn,], y = category, distribution = "multinomial", n.trees = 5000,
shrinkage = 0.01, interaction.depth = 3, n.minobsinnode = 10, nTrain = round(end_trn*0.75),
verbose = TRUE)
#use perf to find best num of trees
gbm.perf(Model)
TestPredictions = predict(object = Model, newdata = all[(end_trn+1):end,], n.trees =
gbm.perf(Model,plot.it = FALSE), type = "response")
TrainPredictions = predict(object = Model, newdata = all[1:end_trn,], n.trees =
gbm.perf(Model,plot.it = FALSE), type = "response")
head(TrainPredictions,n=20)
head(category,n=20)
submission = data.frame(ID = test['imageID'], category = TestPredictions)
submission$max <- apply(submission[,2:15], 1, max)
```

6.5 Testing against hypothese

After training phase, we begin to implement the classification of images randomly picked from our dataset as input for testing. We implement three machine learning models as mentioned before. Therefore, we will have three different output for our testing images. Each training model will generate a csv file to map each testing images to a category. We calculate the correctness of each training model by comparing the human-eye detected category with the machine labeled category.

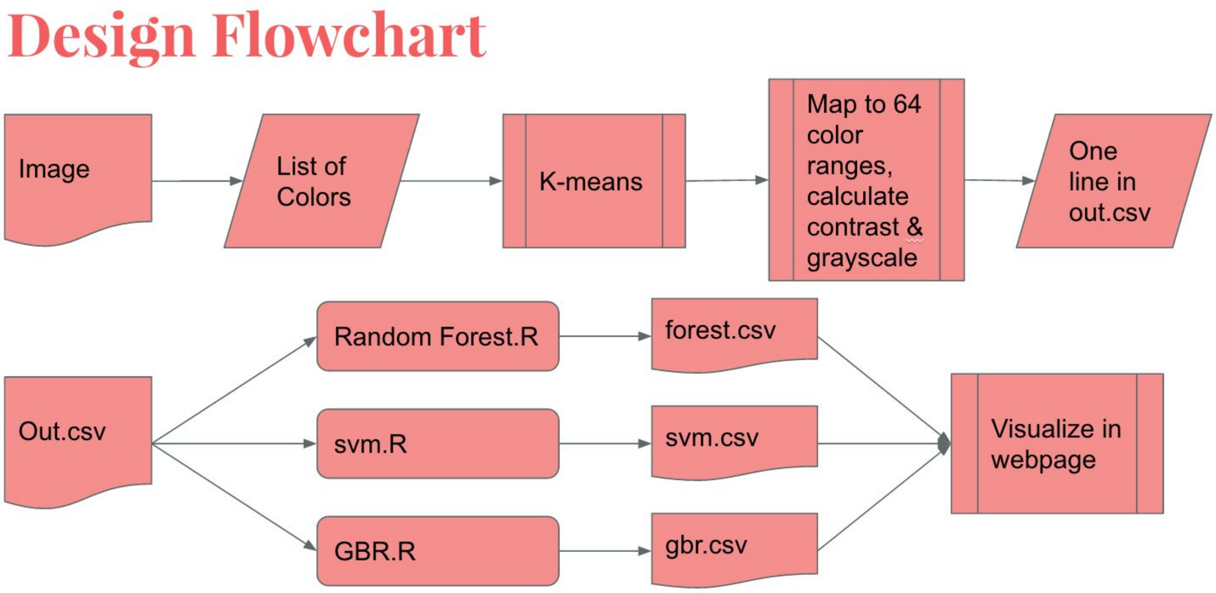
To assure robustness of our tests, we calculated the percentage of correctness by repeating the same test and get the average. We used 75% of our dataset images for training purpose and

25% of our dataset images for classification. We also adjust the k parameters in the Kmeans algorithm. We test the range of k to be (3-8).

6.6 Data Visualization

After the classification, each training model will generate an output to map each of the testing sample image to a specific category. To make the output more vivid and easy to read, we use HTML, CSS to display the images dynamically according to the random testing sample. We will display each category into the same block. Then it is very easy to judge whether the images grouped together into a same category are similar or not.

6. 7 Design document and flowchart



7 Data analysis and discussion

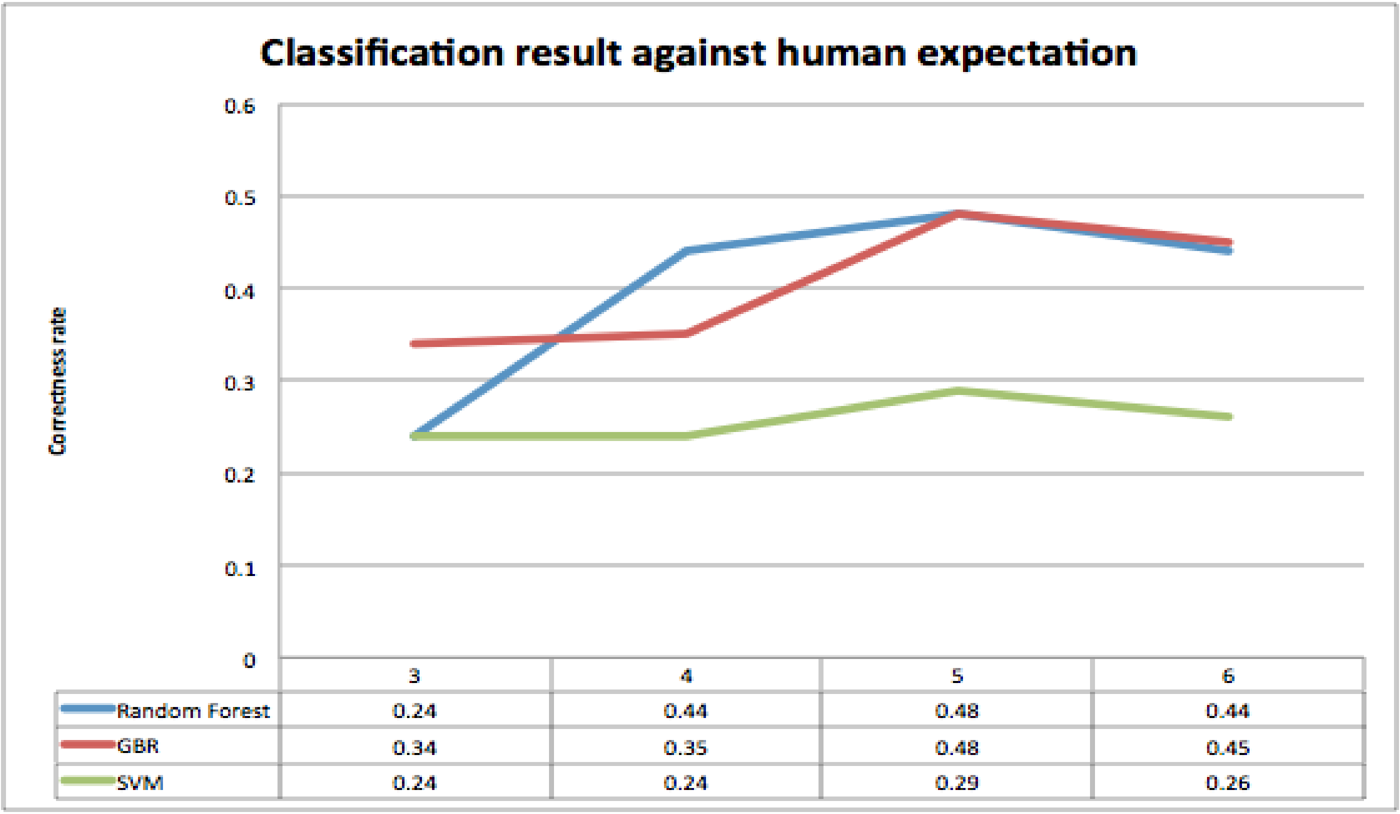
7.1 Output generation

In this study, we first do the image clustering using KMeans to choose the top K colors for each individual images. Then we implement the three learning models and feed the testing samples into the three learning models for the purpose of classification. The following are the results for K = (3, 4, 5, 6) and for the three learning models. 0.24 refers to the correct classification percentage using the learning model against the human-eye classification.

7.2 Output analysis

In the output, we can see that Random Forest and GBR have much higher correct percentage compared with the SVM model. Moreover, it seems that when we choose k equals 5, the classification result seems best. The best matching percentage of our learning models are 48%, which is a pretty high value. We also plotted this result using a line graph to represent the changes of different learning models and different k values.

K	3	4	5	6
Random Forest	0.24	0.44	0.48	0.44
GBR	0.34	0.35	0.48	0.45
SVM	0.24	0.24	0.29	0.26



7.3 Compare output against hypothesis

Generally speaking, this study confirms the two hypotheses of this study. Firstly, it is evident from our classification results that images could be recognized mainly by their colors as styles. This is a very important feature for images because it tells crucial information of image. In the home interior environment, color is more vital in deciding the styles than other features, such as furniture type. Therefore, this study may have great application for home interior search engines and websites, such as Airbnb.

Secondly, the results of classification by our learning models match human-eye labeled styles by 40% on average. We have implemented three learning models. The results for Random Forest and Generalized Boosted Regression match the highest styles, by about 48% when the k equals to 5 in the Kmeans algorithm. The matching percentage for SVM is a little bit lower, which is about 27% on average for different k parameters in the Kmeans algorithm.

7.4 Abnormal case explanation

One of the abnormal case is that our learning models could not differentiate yellow from gold, and could not differentiate yellow to green. Since we use human-eye to do the labeling in the beginning, human-eye is more sensitive to tell the color difference. In our pre-labeling, we have put yellow, green and gold into different categories. However, the learning models is not as sensitive as human-eye in our study. One of the possible reasons is that we mapped all the 256^3 colors into 64 color space. Our color space may be too rough to distinguish light blue from dark blue, and gold from yellow, and other similar colors.

7.5 Discussion

This study explores how to implement image classification based on color style. Results of the three learning models indicate that color is a very important feature to recognize the style of a picture. Also, the random forest model and the generalized boosted regression provide high matching results of classification compared with the pre-labeling by human-eye. Overall, the results of this study confirm our hypotheses that we can use colors to do image classification and also that the learning models are able to learn the image color style and perform classification based on the learning. There are also some abnormal cases as mentioned in 7.4 abnormal case explanation. It turns out that human eyes are more sensitive to vivid and outstanding colors, while machine doesn’t.

8 conclusions and recommendations

8.1 Summary and conclusions

This study aims to classify a collection of images based on styles. In our study, we will define the style of images based on colors, gray scale, brightness, lightness and contrast. There are three stages in our study. The first stage, we extract the features with Java from the images. We also calculated the top k color clusters in each image using KMeans algorithm. Then we map each image to 64 color ranges in color space. In the second phase,we train the model using Random Forest, SVM, Generalized Boosted Regression. In the third phase, we display the classification of images in groups in webpages.

8.2 Recommendations for future studies

This study indicates that colors are very important features for identifying styles for the image. This is especially relevant for home interior design. Therefore, this study is useful for house image search engines. More importantly, the search engine could recommend house images of similar styles based on the customer's previous search and bookmark results.This study also demonstrates that the Random Forest model and Generalized Boosted Regression work very well for learning the styles of the images based on colors.

This study has two limitations: first, the sample of our study (248) is a little bit small for the learning models. In the future, larger sample should be used to test the hypotheses of this study.Second, we used 64 color space to represent all of 256^3 colors. Our color map might be too loose and not catch the minor difference among colors, such as yellow, gold, and green. One abnormal result is that the classification model could not tell the difference between the green and yellow color. After the classification, the images of yellow color and green color are usually mixed together. This result is quite abnormal and out of our expectation. One possible reason is that human-eyes are more sensitive to color difference than the learning models in our study. Another reason as mentioned above is that we may have used too narrow color space to represent all possible colors. In the future, research could try to make the color space bigger so that it can map images to more color space buckets.

Github code https://github.com/susiesu2/ColorStyleClassification.

9. Bibliography

Conci, Aura, and Everest Mathias MM Castro. "Image mining by content."Expert Systems with Applications 23.4 (2002): 377-383.

Dubey, Rajshree, Rajnish Choubey, and Sanjeev Dubey. "Efficient Image Mining using Multi Feature Content Based Image Retrieval System." Int Jr of Advanced Compute Engineering and Architecture 1 (2011).

Gudivada, Venkat N., and Vijay V. Raghavan. "Content based image retrieval systems." Computer 28.9 (1995): 18-22.

Kannan, A., V. Mohan, and N. Anbazhagan. "Image clustering and retrieval using image mining techniques." IEEE International Conference on Computational Intelligence and Computing Research. Vol. 2. 2010.

Zhang, Ji, Wynne Hsu, and Mong Li Lee. "Image mining: Issues, frameworks and techniques." Proceedings of the 2nd ACM SIGKDD International Workshop on Multimedia Data Mining (MDM/KDD'01). University of Alberta, 2001.