Binary Classification of Movie Genres from Posters

MTH441: Linear Regression and ANOVA - Course Project

Course Instructor : Dootika Vats

Sruthi Subramanian

November 20, 2025

1 Introduction

Can we look at a poster and decipher the genre of the movie? As humans, often we can. We tend to relate darker posters with features like ghosts or smoke to horror movies, and bright colorful ones with happy people and maybe cartoons to comedy movies. But can a program in R make this distinction? - without neural networks or any hi-fi machine learning model. Can we do this using just linear regression?

2 Problem Statement

Build a model that classifies any given image of a poster as either 'comedy' or 'thriller'. Scrape your own training data from the internet and build a model that comprises of 4 components:

1. `make_feature()` - a function that takes in one *imager* image and creates a numerical vector of features.
2. `beta_hat` - which is a fitted regression coefficient vector.
3. `class_prob()` - a function that takes the linear prediction($x_i^T \hat{\beta}$) from `beta_hat` and the output vector of `make_feature()`, x_i to produce probabilities of the poster being comedy (class = 0) and thriller (class = 1).
4. `classify()` - a function that does the final classification to class 0 or 1 based on the output of `class_prob()`.

Restrictions: You can use only *imager* in the `make_feature()` function. Other libraries can be used in training and to build the model, but not in testing. Note that the genre is taken to be the first tag of a movie on IMDB.

3 Data Scraping and Cleaning

3.1 Training Data

Source - Movie Genre from its Poster (Kaggle)

I took the poster list and obtained the links to the IMDB pages of each of the movies from this dataset. I had to clean and edit the links as the ones in the dataset are outdated. I then scraped the movie posters from each of the links, and saved them as "Poster" + i + ".jpg", where i is the index of the movie, in the folder - "Posters".

The dataset already had a list of the genres. I cleaned it to take the first tag for each movie. I then created my training dataset, with those having "Comedy" as their first tag making up the training set for comedy movies, and those having "Thriller", "Adventure", "Action" or "Crime" as their first tag for thriller movies. I did this as the dataset I would have gotten by just considering "Thriller", would have been very imbalanced, with many more entries for comedy. Further, as "Adventure", "Action" and "Crime", are very close to "Thriller" in theme, this combination makes sense.

I also shuffled the data I had extracted as it was ordered initially, with all comedy movies coming first, followed by thriller movies.

3.2 Testing Data

Consists of 60 posters manually scraped from the web, 30 for comedy and 30 for thriller. Posters saved in the file "Test dataset". All comedy movies are saved as "comedy"+i+".jpg" and thrillers are saved as "thriller"+i+".jpg", for $1 \leq i \leq 30$.

4 Feature Extraction

```
# Creating feature vector from the images
make_feature <- function(img)
{
  if (spectrum(img)==1){
    img <- imappend(list(img, img, img), "c")
  }
  hsv_img <- RGBtoHSV(img)
  hue <- quantile(as.vector(hsv_img[,,1]), probs = c(0.2,0.4,0.6,0.8))
  saturation <- quantile(as.vector(hsv_img[,,2]), probs = c(0.2,0.4,0.6,0.8))
  value <- quantile(as.vector(hsv_img[,,3]), probs = c(0.2,0.4,0.6,0.8))
  red <- quantile(as.vector(img[,,1,1]), probs = c(0.2,0.4,0.6,0.8))
  green <- quantile(as.vector(img[,,1,2]), probs = c(0.2,0.4,0.6,0.8))
  blue <- quantile(as.vector(img[,,1,3]), probs = c(0.2,0.4,0.6,0.8))
  img_gray <- grayscale(img)
  img_gray <- resize(img_gray, 32, 32)
  img_mat <- as.matrix(img_gray)
  black <- matrix(0, nrow = 32, ncol = 32)
  darkness <- norm(as.numeric(black - img_mat),"2")
  xi <- c(1,hue, saturation, value, red, blue, green, darkness)
  xi <- as.matrix(xi)
  return(xi)
}
```

From the image, the following features are extracted to create the feature vector, x_i:

- Quantiles of the frequencies of hue (the main dimension of color), saturation (intensity), and value (lightness/darkness).

- Quantiles of the frequencies of red, green and blue (the fundamental colors making up the image).
- 'Darkness' - a measure of how dark (or bright) the image is, looking at how different the image is from black.

5 Model - Ridge Logistic Regression

```
class_prob <- function(lin.pred){
  p <-  1/(1+exp(-lin.pred))
  return(p)
}
classify <- function(p){
  if (p <0.5){ return (0) }
  else{
    return(1)
  }
}
```

where lin.pred is the linear predictor, i.e. lin.pred $= x_i^T \hat{\beta}$.

We model the probability of $y_i = 1$ given predictors $\mathbf{x}_i$ using the logistic function as:

$$p_i = P(y_i = 1 \mid \mathbf{x}_i) = \frac{1}{1 + e^{-\mathbf{x}_i^\top \boldsymbol{\beta}}}.$$

We predict $\mathbf{0}$, if $p_i < 0.5$, and $\mathbf{1}$ otherwise.

The glmnet library allows us to find the optimal $\hat{\beta}$ for this model with penalisation of a ridge model, which we then use for prediction.

6 Libraries Used

1. *stringr* - for data cleaning
2. *rvest* - for web scraping
3. *imager* - for processing images
4. *glmnet* - for training, i.e. finding the optimal `beta_hat` for ridge logistic regression

7 Files

Data Scraping and Training.R

Contains codes for data scraping, data cleaning, creation of training dataset from scraped data, the model and training. "Model.Rdata" is saved from here.

Model.Rdata

Contains the model, including the functions - `make_feature()`, `class_prob()` and `classify()`, and the vector `beta_hat`.

Testing.R

Contains codes for testing the model, using the test dataset (posters from the folder "Test Dataset").

MovieGenre.csv

Csv file obtained from Kaggle with list of movies, genres and IMDB codes.

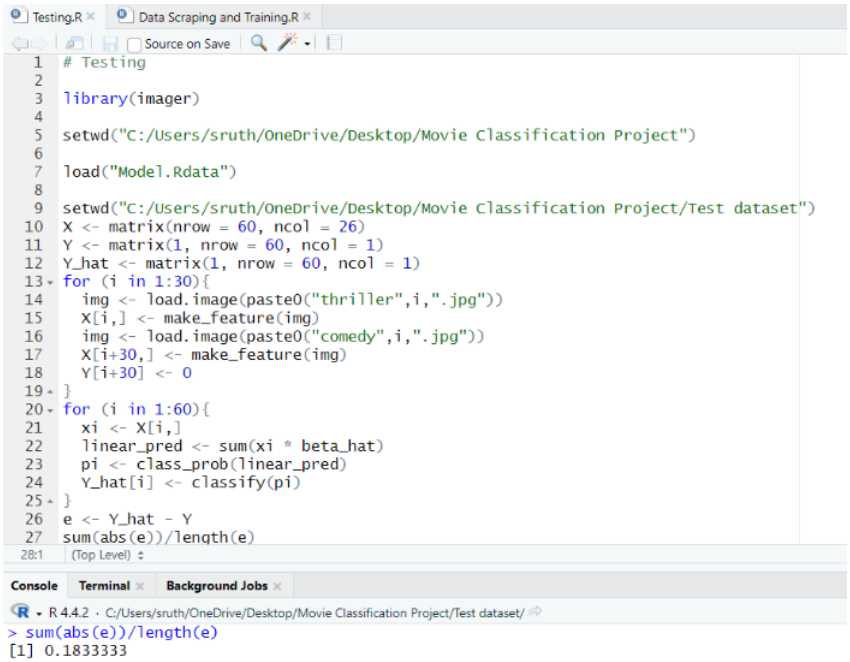
Posters (Folder)

Contains all scraped posters for training.

Test Dataset (Folder)

Contains all scraped posters for testing.

8 Results from Testing



As seen above, 81.67% accuracy was obtained on the testing set!

9 Reproduction Guidelines

- All posters for training can be scraped by running the respective code in "Data Scraping and Training.R"
- All codes can be run on one's local system once the respective paths (in setwd("-path-")) are changed accordingly, to reproduce results.

10 Limitations

- The features extracted are not necessarily sufficient to describe the image. Exploring spatial properties, like using a method that extracts objects in the image, possibly via clustering, may prove to be useful.
- The training data merges action, crime, and adventure also with the thriller data set. This may not be the best way to address class imbalance. Instead, different ways of sampling may be explored.
- Other link functions like the log-log link or other models like LASSO can be tried out.