Nicholas Lechner

1/12/2016

**Hedgeable Machine Learning**

**Predicting Image Advertisements in R**

**Can a machine learning model be used to predict whether a given record is an advertisement or not?**

**Hypothesis:** We will test two machine learning methods: **Decision Tree Classification**, **Ordinary Least Squares Linear Regression,** and **Logistic Regression** against 2 formulas to determine whether a given image is an advertisement.

Formula 1:

AD ~ Height + Width + Local

Formula 2:  
AD ~ ARatio + Local

\*aratio = height:width or width:height ratio where the numerator is the larger of the two

Given that advertisements tend to take the same shape and size on most websites, these attributes can reasonably be expected to predict whether an image is an ad.

**Method:**

Use conventional validation to test both formulas against the Decision Tree, OLS, and Log Models averaged over 100 samples and determine which model more accurately predicts whether a record is an advertisement, using 70% of the data as the training set and 30% as the test set.

Using a for loop: generate train and test sets to run each model against 100 times and test against three different parameters to determine the effectiveness of each model.

Parameter 1: Number of total predictions accurately predicted over the total test size

Parameter 2: Number of **total** advertisements predicted over the actual number of ads in the test data set

Parameter 3: Number of ads **correctly** predicted against the actual number of advertisements in the test data set - this will be the best indicator of model performance

**Attributes and assumptions:**

There are 458 advertisements and 2820 non advertisements in the known dataset, meaning our model will ideally predict about 14% of the test data to be advertisements.

There are 910 instances of individual images in the data which are missing their respective continuous variables: height, width, and a-ratio. Given that both formulas include at least one of these attributes, the missing values for height and width must be imputed and their respective a-ratio calculated. This is how we will handle the unknown cases.

We impute the missing height and width values using k-Nearest-Neighbor Imputation from R’s “VIM” package with the number of nearest neighbors k set to 5.

Applying the kNN algorithm to the dataset as a whole imputes the missing values of H and W based on the 5 nearest neighbors in the data which share the same attributes for the other columns, ie the url, origurl, ancurl, and alt content. Images sharing these features can be expected to share the same image size. From there, the images respective a ratio is calculated based on whether the height or width is the greater.

**Testing the Models:**

In order to test the models side by side, I wrote a function which takes the data frame, in this case the read.csv() input of data.csv after the kNN imputations and a-ratio calculations were performed, and the desired train subset size (70% of the data) as inputs. The data frame is shuffled using the sample() function and the data frame is split at the 70% mark to generate the train and test set. This shuffling procedure allows us to test each of the 3 models on entirely new unique training and test set every time.

We then apply the 3 models we are testing on the TRAIN data frame set using R’s lm(), glm(), and rpart() functions, afterwhich we use R’s predict() function applied to the TEST data frame, which generates a predicted value for whether an image is an advertisement or not based on the model and formula specified. The predicted values for each of the three models are rounded to either 0 or 1 at a less than or greater than .5 cutoff to indicate whether our model has predicted an image is a non advertisement or an advertisement. This procedure is done 100 times for each of the 3 models in each round so that the same TRAIN set is used to build each of the models and they are tested against the same TEST set.

**Results:**

The summary statistics for each parameter and formula are as follows

**Formula 1: AD ~ Height + Width + Local**

|  |  |  |  |
| --- | --- | --- | --- |
| **MODEL** | P1:  # of accurate pred / total # images | P2:  # of pred ad / total # of ad images | P3:  # of correct ad pred / # of total ad images |
| OLS Linear | Min. :0.8993  1st Qu.:0.9145  Median :0.9186  Mean :0.9195  3rd Qu.:0.9250  Max. :0.9349 | Min. :0.5500  1st Qu.:0.6145  Median :0.6475  Mean :0.6491  3rd Qu.:0.6876  Max. :0.7917 | Min. :0.4444  1st Qu.:0.5074  Median :0.5328  Mean :0.5369  3rd Qu.:0.5601  Max. :0.6345 |
| Logarithmic | Min. :0.8983  1st Qu.:0.9135  Median :0.9186  Mean :0.9181  3rd Qu.:0.9237  Max. :0.9329 | Min. :0.5674  1st Qu.:0.6213  Median :0.6547  Mean :0.6592  3rd Qu.:0.7020  Max. :0.7917 | Min. :0.4444  1st Qu.:0.5074  Median :0.5328  Mean :0.5372  3rd Qu.:0.5601  Max. :0.6414 |
| Decision Tree Classification | Min. :0.9217  1st Qu.:0.9346  Median :0.9395  Mean :0.9393  3rd Qu.:0.9433  Max. :0.9552 | Min. :0.5594  1st Qu.:0.7113  Median :0.7581  Mean :0.7591  3rd Qu.:0.8019  Max. :0.9787 | Min. :0.5524  1st Qu.:0.6328  Median :0.6607  Mean :0.6627  3rd Qu.:0.6904  Max. :0.7833 |

\*how to read the table:

Each box contains the summary statistics for the given model and parameter. In the case of P1 OLS, the OLS model predicted whether 91% of the images were advertisements or not correctly.

**Formula 2: AD ~ Aratio + Local**

|  |  |  |  |
| --- | --- | --- | --- |
| **MODEL** | P1:  # of accurate pred / total # images | P2:  # of pred ad / total # of ad images | P3:  # of correct ad pred / # of total ad images |
| OLS Linear | Min. :0.8291  1st Qu.:0.8484  Median :0.8545  Mean :0.8551  3rd Qu.:0.8616  Max. :0.8881 | Min. :0.009174  1st Qu.:0.033602  Median :0.043798  Mean :0.046937  3rd Qu.:0.057628  Max. :0.108108 | Min. :0  1st Qu.:0  Median :0  Mean :0  3rd Qu.:0  Max. :0 |
| Logarithmic | Min. :0.8291  1st Qu.:0.8484  Median :0.8540  Mean :0.8550  3rd Qu.:0.8616  Max. :0.8881 | Min. :0.009174  1st Qu.:0.033602  Median :0.043798  Mean :0.047890  3rd Qu.:0.057881  Max. :0.109375 | Min. :0  1st Qu.:0  Median :0  Mean :0  3rd Qu.:0  Max. :0 |
| Decision Tree Classification | Min. :0.9034  1st Qu.:0.9207  Median :0.9247  Mean :0.9256  3rd Qu.:0.9301  Max. :0.9461 | Min. :0.4326  1st Qu.:0.5026  Median :0.5404  Mean :0.5457  3rd Qu.:0.5968  Max. :0.7252 | Min. :0.4184  1st Qu.:0.4698  Median :0.5000  Mean :0.5040  3rd Qu.:0.5411  Max. :0.6172 |

**Findings:**

For formula 1, the OLS, Log and Decision Tree models performed similarly across parameter 1. For parameters 2 and 3, the Decision Tree model outscored both the a margin of 15% and 10% respectively.

For formula 2, the Decision Tree model outscored the OLS and Log models by about 7-8% for parameter 1. For parameters 2 and 3, the Decision Tree model outscored the OLS and Log models by almost 50% in both cases - as neither the OLS or Log models were able to predict whether a given image was an advertisement based on the a-ratio. This leads to the conclusion that the OLS and Log models exclusively predict the presence of a non-advertisement when the a-ratio is considered. We declare the Decision Tree model to be superior in both formula cases.

**Conclusion:**

A decision tree classification tree is prefered to a OLS or Log regression model to accurately predict whether a given image is an advertisement or not. The model should be trained on the entire images data.csv.

A new test data set where the presence of an advertisement is not known could be generated by a web crawler in R-Selenium which scrapes image height and width and then calculates the a-ratio across multiple web pages. Unknown heights and widths should be imputed using kNN Classification based on url content and other attributes. The decision tree model can be refined using random forests.

**Relevant Files:** A description of the files included and what they do:

cleaning.R

Imports data.csv and cleans the file. Imputes na values with kNN imputation. Generates a unique ID for every image.

functions.R

Contains a preliminary test\_model() function used to evaluate parameter 1, build\_test\_set() which shuffles and creates train and test subsets, and return\_result() which was a preliminary function used to pass formulas through rpart()

generate\_predictions.R

File used to generate predictions with the OLS, Logarithmic, and Decision Tree Models. Predictions for each of the 3 models are cbinded() to the image IDs and whether the image was actually an ad or not.

testing\_paramaters.R

Contains the functions used to evaluate the model across the 3 parameters

100\_Tests.R

Contains script used to test the 3 models over a for loop 100 times and calculate the summary statistics of the given 3 parameters.

plots.R

Contains scripts used to generate graphs

**Graphs and Images:**







