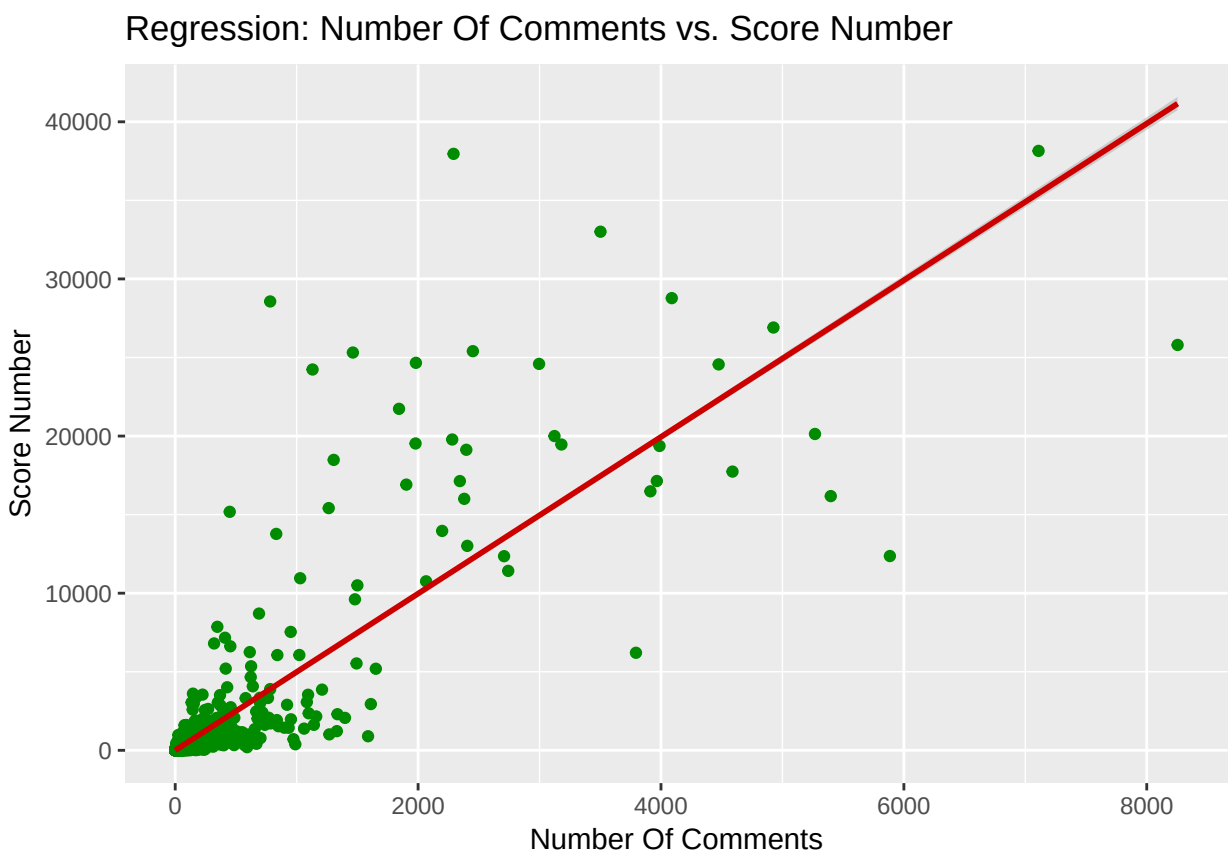
Datascience Fundamentals with R - Reddit

Group 7

December 5, 2017




```
##
## Call:
## lm(formula = D$score ~ D$num_comments)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16979.2      0.4      0.4      0.4  26532.1
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.61964    4.85125   0.128    0.898
## D$num_comments  4.98617    0.02564 194.443   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 599.7 on 15398 degrees of freedom
## Multiple R-squared:  0.7106, Adjusted R-squared:  0.7106
## F-statistic: 3.781e+04 on 1 and 15398 DF,  p-value: < 2.2e-16
```

Goal

Analyze reddit news posts because they give us a lot of information about what is important to society at a particular time. We use reddit because it is one of the most popular websites in the world and has a very active user community, with 1.6 billion users. The news subreddit alone has 70,000 posts in a month.

We believe that the posts of the top authors (authors that have the most posts) are representative of the community since their content should be varied. Furthermore, given the variety of news posts and the sentiments associated with them (there is always good & bad news) we think that there should be a correlation between good news and positive sentiments and bad news with negative sentiments.

We can assume that a particular author may lean towards a more positive or negative sentiment - we thus assign a variable `author_score` to represent how positive or negative a particular author is. This measure is calculaetd as the *summed sentiments of all posts of this particular author* , the question now becomes:

For each post, can we determine whether their post will be well liked, given the author's general sentiment?. Our hypothesis is that posts with a positive sentiment will be received better than those with a negative one, so authors that have a generally positive sentiment would have much more upvotes (positive correlation between `author_score` and `like_score`)

Data

We decided to use one month of Reddit News data. We picked December 2016 for this as it was just after the US elections and that enough time has passed for us to (possibly) notice the effects of news that came out at that time.

Reddit news data usually comes in the form of a title and a link, the title is written by the user and links to the news website's page. It sometimes has thumbnails.

Reddit post data is available on google's bigquery database, it is saved in `news_2016_12.csv`. Following are the most relevant columns for our analysis:

- `time_created` (UTC timestamp) - when was the post created
- `author` - username of user that posted
- `domain` - which domain did the news come from?
- `url` - Specific URL of the news post
- `score` : upvotes - downvotes (renamed as `like_score`)
- `upvotes` : how many "likes" the post received
- `downvotes` : howmany "dislikes" the post received
- `title` : user-created title of the post

Due to an issue with the API, no downvote data is given to us. Thus the `score` is equal to the `upvotes` of a post. We will be using the `score` field moving forward.

Furthermore, we thought it would be interesting to compare the difference between the user-generated title and the Actual title posted by the news agency. For this, a (scrapy)[https://scrapy.org/] spider was created to crawl all the (cleaned) URLs and retrieve the title. We think this was rather successful since it retreived `24,045` titles form about `31,713` cleaned posts. This is saved as `titles.csv`.

Task

From the data above, we want to do the following:

1. Discover the relationships and trends between elements in our dataset (fig 1)
2. Acquire the sentiments of words and titles via NLP
3. Classify the sentiment of the titles of the most popular authors.

Due to the size and variety of elements in our dataset, we will apply a variety of methods to extract information, and may use external data to improve our analysis. However, this also implies that the performance measures of our ML algorithms will vary with the specific relationship under examination.

Cleaning `[clean.r]`

Everything related to cleaning the original dataset is defined in `clean.r`. Essentially, the script:

- saves `utc_created` as a POSIXct variable
- removes special characters from titles
- removes non-english titles, which halves our original dataset.

Getting relevant data - `[05_Classification_Data.R]`:

1. We remove all authors with the value `[deleted]`, because this is value given to users who have deleted their accounts, which mean that we become unable to different specific users.
2. We only take posts who have a `like_score > 1`. Since all posts created automatically have a score of 1, this does not really tell us anything about the correlation of upvotes and downvotes. (fig 1) shows that the long tail of the distribution is when the `like_score = 1`

After these two operations, our dataset is reduced to `3946` rows compared to the originial `31,713`, and this is further reduced to `394` upon generating the `author_score`

We split the data set into `80% testing; 20% training`.

Descriptive statistics and analysis

General Statistics:

Here we start our examination of the dataset by looking into the relationships between the different features we have, for example:

1. The distribution of upvotes and comments
2. The correlation between upvotes and comments
3. The distribution of posts over time

Distributions

How are the upvotes / comments distributed?

We use take `log(score)` to reduce the scale of the graph (posts with `score=0` are mapped to `score=-1` to avoid a `-inf` result). We can clearly see that there is an exponential distribution and it is interesting to note that the longest tail is where `score = 1`

The comments follow this distribution

Correlation between upvotes & comments

Distribution of posts over time.

There is some cyclicality of posts during both the month and the day. This may have implications on upvote score depending on time/date posted. Our paper will not focus on this.

Sentiment - NLP with `tidytext`:

We take the sentiment of each word found in our training dataset with 3 different libraries - bing, mcr and afinn. Afinn is the most useful in these classifications since it assigns a weight between `-5 < weight < 5` for every word. This is the algorithm we use for the classification algorithms that we will present later.

The new Dataset

Machine Learning

Overview

We used 4 different supervised classification algorithms to examine what we can learn from this dataset. One of the core tasks of machine learning is classification of data points.

Reddit, being one of the most popular websites on the internet has a very wide reach. We think that being able to classify a post on whether it is good or bad is the first step in being able to determine/study how to make a good post. Since we are analyzing a news dataset, the value this brings is to easily determine what topics are relevant to people at a certain time period, based on the assumption that good posts will generate more upvotes and/or comments.

Thus, for each algorithm, we will outline the following:

1. An overview of the algorithm
2. Task: What is the task to be solved, and why is it relevant?
3. Experiece: How does the training look?
4. Performance: By what measures to we determine the success or failure of our algorithm?
5. A discussion on the implementation

But before we go to ### 1. Perceptron Classification

The perceptron is the simplest neural network one can build. Because we have a big dataset, we will apply it via a Stochastic Gradient Descent algorithm. It is based no the concept of a simple brain, where it fires if the signals it receives are above a threshold, and stays dormant if this threshold is not reached.

For this algorithm, we want to classify the sentiment (positive/negative) of a post given 2 features - (1) Author's sentiment, (2) Post's score

2. Logistic Regression

Despite being called a "regression", the logistic Regression is a classification algorithm, instead of a predictive algorithm. The concept is based on the logistic function, whose output is in the range between 0 and 1.

3. Bayes Classifier

The bayes classifier is a classification algorithm based on probability theory. The idea is to get the class where a certain value *most probably* belongs to. Here, we maximize the likelihood.

4. Neural Network