**Capstone Project**

Ross Beck-MacNeil  
November 22nd, 2017

**1 - Definition**

**1.1 - Project Overview**

Machine learning has been successfully used to classify documents by topic for several decades (Sebastiani 2002). However, machine learning techniques do not perform as well when performing sentiment analysis which requires the parsing of more complex language structures (Bo et al. 2002). This is where ideas from natural language processing must be applied. Humour is even more subjective and is harder to analyze than sentiment.

My aim with this project is to make progress towards solving a complex natural language processing problem using machine learning. I think that it would very beneficial if computers could interpret and produce the same kind of natural language of which even young children are capable. This would allow machine learning to be applied to a wider variety of tasks than it is currently capable of solving. In particular, machine learning could be applied to many problems that are not well structured and for which there is not much training data.

**1.2 - Problem Statement**

I will attempt to solve the problem of determining whether a post on the reddit r\jokes subreddit has received more than 10 net upvotes, conditional on its title and submission text. The criteria of 10 net upvotes can be considered a proxy for the true metric of interest, the funniness of the post. However, humor is very subjective and hard to quantify and therefore the net upvotes from the r/jokes subreddit is being used as a proxy for the funniness of the post.

This is a problem that reoccurs every hour of every day, as users are constantly posting new submission to r/jokes. Not only is the r/jokes subreddit constantly producing a new stream of problems to be solved, there is ample historical data so that a model can be trained and learned.

This will be a supervised binary classification problem and as such many different measurable evaluation metrics could be applied. These will be further elaborated in *Section 6 - Evaluation Metrics*. Given the title and text of a post, the solution will be a machine learning model that receives as input the text and title of the post and outputs either a class prediction or a class probability. The solution will be further described in *Section 4 - Solution Statement.*

**1.3 - Metrics**

Given that this is binary classification problem, there are many different metrics that could be used to evaluate the solution. The most common metric is accuracy. If is the true value of observation i, such that

.

and is the predicted value of the *i*-th observation, then the accuracy of the predicted values can be calculated as:

where *N* is the number of samples and is the indicator function such that:

However, the accuracy metric can be misleading in certain circumstances. For one, it can be hard to interpret accuracy when one class occurs much more frequently. This is known as an unbalanced classification problem. In addition, it does not account for the confidence that the classifier has in its predictions. It seems reasonable to prefer a model that is more confident in its correct predictions and less confident in its incorrect predictions.

The Area Under the Receiver Operator Curve (AUROC) metric can overcome the shortcomings of the accuracy metric. The Receiver Operator Characteristic (ROC) is a graphical plot that illustrates the classification ability of a binary model as its discrimination threshold is varied. The area under this curve (AUC) is the probability that the classifier will produce a more confident positive prediction for a randomly chosen positive example than a randomly chosen negative example.

The one drawback to the AUROC is that it can only be calculated for algorithms that output confidence scores. Some algorithms such as support vector machines and k-nearest neighbors do not natively output such confidence scores. Therefore, this project will focus on models that natively produce confidence scores.

It is important to emphasis that proposed solution will be ultimately evaluated using its predictions on the hold-out test set. It is quite easy to develop a classifier that achieves perfect accuracy or AUROC on data that it has seen. It is quite another task to develop a classifier that does well on unseen data. However, care must be taken not to evaluate too many potential solutions against the hold-out set. Ideally model and hyperparametric selection will take place using cross-fold validation or a validation set that is different from the test set. Using the test set for these purposes could lead to overfitting in the same way as using to directly optimize the model parameters

**2 - Analysis**

**2.1 - Data Exploration**

**2.1.1 – Reddit Jokes**

The primary dataset that is being used contains all posts from the Reddit subreddit r/jokes that were submitted between January 25, 2008 and September 10th, 2017. I personally obtained this dataset by using the reddit API to query and retrieve these submissions.

The input dataset is in comma separated values (csv) format and is entitled *AllSubsFrom\_rJokes\_14\_09\_2017.csv*. The 14\_09\_2017 indicates the date that the scraping finished: September 14th, 2017. The data set, before any cleaning or processing, contains 319,747 observations. Each observation corresponds to a different reddit post. There are a total of 9 variables in this data set and they are described in Table 1.

Table

|  |  |
| --- | --- |
| **Variable** | **Description** |
| *id* | Unique identifier assigned by Reddit |
| *date* | Post submission date, in Unix epoch time (seconds since January 1, 1970) |
| *downs* | Number of downvotes. Due to reddit protections, always equal to 0 |
| *score* | Net upvotes (fuzzed) |
| *text* | Text of submission/post |
| *title* | Title of submission/post |
| *ups* | Number of upvoter. Due to reddit protections, this is equal to the score variable |
| *upvote\_ratio* | Upvote ratio, ratio of upvotes to total votes |
| *url* | Link to original post. |

Given the scope of the problem, it is very appropriate to use this data set, as it conforms exactly to the problem statement from section 2. Per the specifications of the problem statement, a new *funny* variable will be created such that for *ith* observation:

Jokes with 1 < score < 10 are considered neutral, neither funny nor unfunny, and will not be considered by the model and will dropped from the dataset. This approach is similar to the one employed by Bo et al (2002) where they focus on discriminating between positive and negative movie reviews and do not consider neutral ones. This *funny* variable will be the target variable that the solution will attempt to predict. At times it may be referred to as *y*.

After removing neutral posts, 12.5% of the observations will be randomly selected into a holdout test set. This will the set of observations upon which the final model will be evaluated. The models will be developed only against the remaining 87.5% of observations. This separation between the train and test sets is very important.

**Some examples of the jokes….**

**2.1.2 – Pretrained Glove Embeddings**

In addition to the dataset containing the reddit posts, datasets consisting of pretrained GloVe word embeddings will used. These datasets are available from <https://nlp.stanford.edu/projects/glove/> (Pennington et al. 2014). Each dataset contains one line per word, where each line starts with the word and is followed the elements of its vector representation. The word and its elements are separated by spaces. In order to test the effect of using vector embeddings of different dimensions, three of these pre-trained word embeddings files will be used. The data sets that will be used will correspond to vectors of 50, 100 or 200 dimensions. It is expected that the pre-trained embeddings will allow for the development of models that account for semantic similarities while avoiding overfitting to the relatively small training set.

**2.2 - Exploratory Visualization**

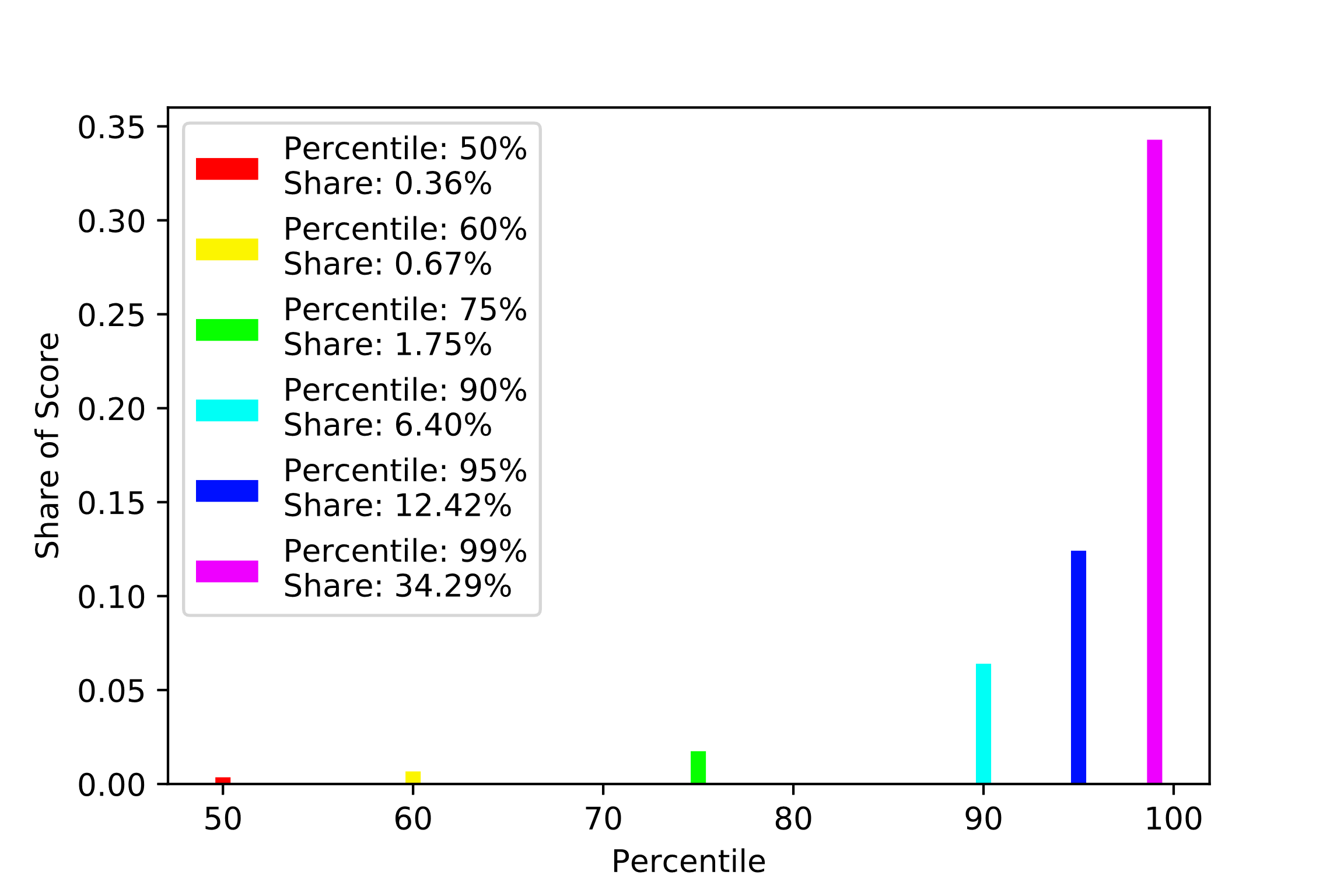
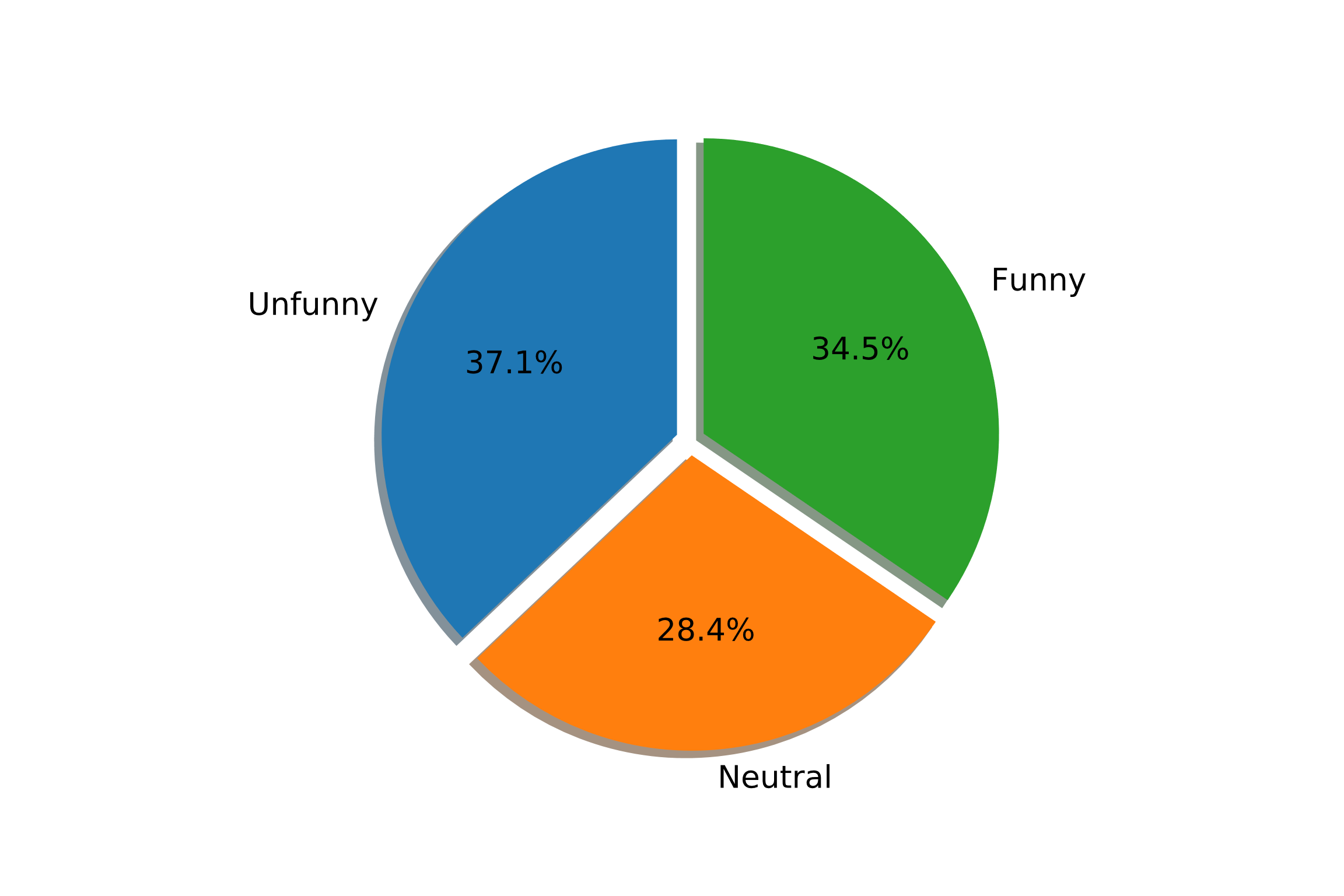
Figure \_\_\_ below, supports the decision to turn this into a binary classification problem, rather than a regression. It show that the distirbution of scores is very uneven.. The top 1% of posts account for obver 65% of total upvotes.****

Figure \_\_\_ below, supports decision to classify posts as funny if score greater than equal to 10, unfunny if 0 or 1, and to drop if between 1 and 10. Very even breakdown.



**2.3 - Algorithms and Techniques**

* *Are the algorithms you will use, including any default variables/parameters in the project clearly defined?*
* *Are the techniques to be used thoroughly discussed and justified?*
* *Is it made clear how the input data or datasets will be handled by the algorithms and techniques chosen?*

I will be employing three classes of algorithms to solve this problem: linear models, decision trees and neural networks. The benchmark linear model and decision tree models are “traditional” machine learning models and will act upon the same transformed data. The neural network model will take the data in a slightly different way.

A linear logistic regression will form my benchmark. This is because there are not very many hyperparameter to tune for linear logistic regression:

* regularization strength; and
* regularization type

For linear regression models, regularization strength is determined by term commonly called either *lambda* or *C.* When regularization strength is expressed as lambda, a higher value implies more regularization. The more a linear model is regularized, the more it prefers coefficients that are closer to 0. C is simply the inverse of lambda. I will use a grid search to determine the optimal C value for the benchmark linear model.

In addition to regularization strength, the only other hyperparameter that a linear regression model requires is regularization type. This refers to how the distance from 0 of the coefficient is calculated. The l1 or l2 norm can be used, or a mix of the two. I will use l2 norm, since it generally provides better results. L1 regularization produces a sparse coefficient matrix. This can help with interpretability but I am not concerned about this.

**2.3.1 – Linear Logisctic Regression**

* Regularization
  + Strength
  + L2 or l1
* Solver….

**2.3.2 – Decision Trees**

* **Basic**
  + Max\_depth = None
  + Min\_sample\_split = 2 (also min\_wighte rferacvint leaf, and min\_sample\_leaf)
  + Using min\_)sample\_leaf and min\_imnpurit split
* **Random Forest**
  + Max\_features
  + And n\_estormators
  + Also has same regularization paramters as basic but not needed or used
  + Regularization comes from the max\_fatues and n\_esimtaores (bagging**)**

**2.3.3 Neural Networks**

**2.4 - Benchmark**

I will use the results of the linear model as the benchmark. This is because linear regression is to train and predict and scales well.

More “advanced” technqies should beat it. Otherwise, what is the poiunt?

# III. Methodology

### Data Preprocessing

1. Drop too short length
2. Create target variable and drop netural
3. For linear and decision trees
   1. Bag of words
      1. Sklearn bulti in preporcesingf (accents and punct stipping)
      2. Sklearn built in tokenizer
      3. Sklearn enlgih sotpwords
      4. Min doc frequency: 5
      5. Max doc share: 0.8
   2. Tfidf weighing
   3. Feature selection through best linear model
4. For neural
   1. tokens
   2. Embeddings
   3. Masking
   4. min

**Implementation**

* Linear model
  + Sklearn
  + Best C valuie and number of features found using 8 fold cross validation – grid seaerch (aacuracy)
* Decision tree
  + Basic
    - Sklearn
    - Best min\_impurity\_decrease and min\_samples\_split found using Random serch, 8 fold cross validation
  + Random forst
    - Skelanr
    - Best n\_estomators and max\_featurs forund using out out bag error.
* Neural networks
  + Embeddgin layer
  + Recrrubnet layer
  + Dense
  + Dense (output

**Not hdiddnetRefinement**

|  |  |  |
| --- | --- | --- |
| **Model** | Optimized? | Params |
| Linear | No | C = 1 |
|  | Yes | C = 16867 |
| Decision Tree | No | param\_min\_impurity\_decrease = 0  **param\_min\_samples\_split = 0** |
|  | Yes | param\_min\_impurity\_decrease =  0.000144  **param\_min\_samples\_split = 7** |
| Random Forest | No | Max\_features = ‘sqrt’  N\_estimators = 10 |
|  | Yes | Max\_features = ‘sqrt’  N\_estimators = 10 |
| Neural | No | regularizers.l2(0)  EMMBEDING DIM?  FULLY\_CONNECTED LAYERS? |
|  | Yes |  |
|  |  |  |

**IV. Results**

**Model Evaluation and Validation**

**Justification**

**V. Conclusion**

**Free-Form Visualization**

In this section, you will need to provide some form of visualization that emphasizes an

**Reflection**

In this section, you will summarize the entire end-to-end problem solution and discuss

**Improvement**

**Before submitting, ask yourself. . .**

* Does the project report you’ve written follow a well-organized structure similar to that of the project template?
* Is each section (particularly **Analysis** and **Methodology**) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?
* Would the intended audience of your project be able to understand your analysis, methods, and results?
* Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes?
* Are all the resources used for this project correctly cited and referenced?
* Is the code that implements your solution easily readable and properly commented?
* Does the code execute without error and produce results similar to those reported?