Analyzing YouTube Channel Data with Linear Regression

Topic Description

The dataset I chose, located here, is a set of YouTube channel metrics from a company called Socialblade. It contains the 5000 top YouTube channels as picked by Socialblade, and my goal is to use this dataset to predict the number of subscribers a given channel has.

Data Description

The data from Socialblade came in CSV format with the following columns:

- `rank` : A string value which represents the Socialblade ranking. Strangely, this is in a format like '1st', '2nd', '3rd', etc.
- `grade` : A letter grade which seems to also represent the Socialblade ranking. It is in a format like 'A++', 'A+', 'B-'.
- `channel name` : The name of the YouTube channel, which is useless for the purposes of predicting subscriber count.
- `video uploads` : The number of videos uploaded by the channel.
- `subscribers` : The number of channel subscribers.
- `video views` : The number of total video views across all videos on the channel.

This data is useful, but needs to be modified in order to be used in our linear regression model. I modified this dataset in a few ways:

- Remove all non-numeric characters from the `rank` field in order to get a proper integer value for the rank.
- Map the `grade` field to an integer value. In the case of this data, there were only 5 unique grade values, so they were assigned a numeric value from 1 to 5.
- The channel name field was removed entirely.
- The subscribers number was moved to be the last column because it will be our target data.

This conversion was done using a ruby script:

```ruby
#!/usr/bin/ruby

require 'csv'

data = CSV.read('data.csv')
headers = data.shift

grade_map = {
    'A++' => 5,
    'A+' => 4,
    'A' => 3,
    'A-' => 2,
    'B+' => 1
}

CSV.open('modified_data.csv', 'wb') do |csv|
    csv << ['Rank', 'Socialblade Grade', 'Number of Uploads', 'Video Views', 'Subscribers']
    data.each do |entry|
        next if entry[0].empty? or entry.join.include?('--')
        csv << [entry[0].gsub(/\D/, ''), grade_map[entry[1].strip], entry[3], entry[5], entry[4]]
    end
end
```

Method

To perform the linear regression, I used the `LinearRegression` model built into ScikitLearn. By splitting our source data into data and target, and feeding it into the model, performing a regression and fit is as simple as:

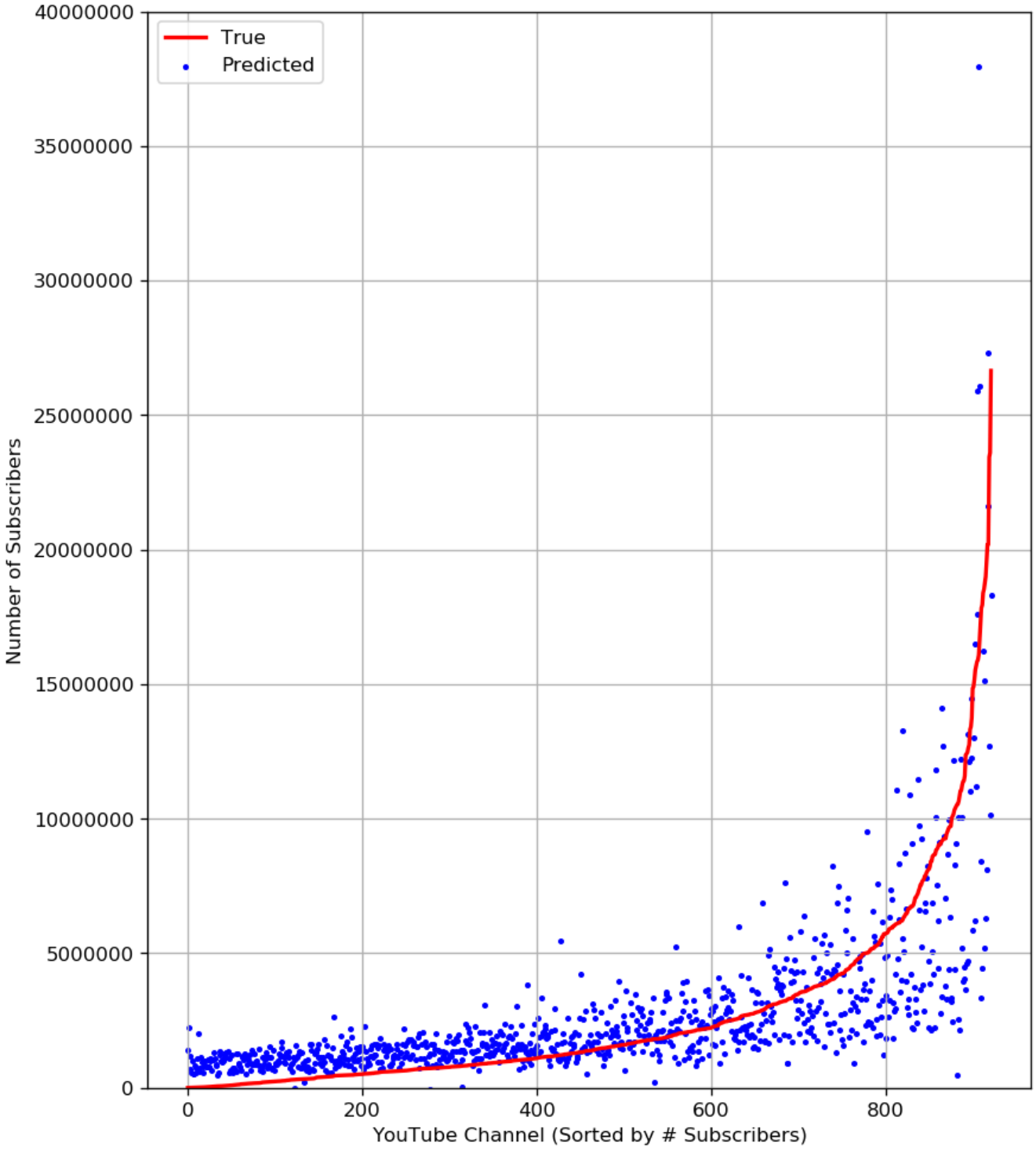
```
regr = linear_model.LinearRegression()
regr.fit(train_x, train_y)
pred_y = regr.predict(test_x)
```

Out of 5000 total data points, I chose a test size of 20%. Additionally, after regression, the loss function was evaluated for both the training and testing datasets. Values were then plotted using matplotlib.

Results

The results of the regression and fit are:

```
Coefficients:
 [-2.12550842e+02 -2.41329858e+04 -1.55060352e+01  1.49052923e-03]
Intercept: 1553929.0240461845
Training Loss Function: 2851165702066.768
Test Lost Function: 2689191972012.4424
```



The loss function values clearly show how poor this model is. In fact, this model is effectively useless. I have a feeling that increasing the size of the dataset might help a bit, but I think that the bottom line is that the data I've provided is just not as correlated with subscriber count as one would expect.

Conclusion

The resulting model fits extremely poorly, and we could benefit from doing some research as to what other features we could add to the dataset.