Model to Predict Overall Score-  
DC Comics Superheroes

[Apoorv Akhouri, MS BIA]

[Stevens Institute of Technology]

Date: 12/16/2020

**Abstract**

As kids we have always wondered who would win in a battle between Superman & Batman, or between Flash & Green Arrow, or even between the Gods: Superman & Wonder Woman. We would collect & trade cards to keep ahead of our friends having the best possible set of superheroes in our pockets.

Later these debates started accounting for different versions of each superhero too as they were introduced in comic books, animated series’ or movies. This creates a lot of confusion on questions like, who will win between Christian Bale as Batman & Ben Affleck as Batman. What would be the deciding factors? Etc.

Over time, people started having villains as their favorite characters too. Bane from the batman series is one of the most adored villains by the fans, due to his strength & the fact that he has defeated Batman on various occasions, & if we are talking about villains, Joker has always been a fan favorite. If you notice, the Joker doesn’t have super strength, but he has still defeated Batman on uncountable number of instances (& the number of versions Joker has is a whole different research topic). So how does one really know who is better than whom?

To be honest, the ‘fictionality’ of these superheroes makes it impossible to ever successfully determine who will win or lose. But what we can do is rely on numbers, statistics & data to give us a clearer picture.

**Model to Predict Overall Score-  
DC Comics Superheroes**

Here we have a data set that carries all the information for about 1400+ superheroes, their history, their characteristics, their individual scores for strength, speed, durability etc. & information on the particular powers they possess.

We will build a model that will predict the ‘overall score’ of these superheroes as accurately as possible, based on the characteristic scores that they have. This will enable us to keep track of any new superhero (or another version of an existing superhero) that is introduced to the audiences, provided we get to determine his/her/their characteristic scores, & almost instantly predict his/her place in the Superhero World. In this model, we would be focusing only on DC Comics Universe & its superheroes.

# Data & its Source:

The data was retrieved from [www.kaggle.com](http://www.kaggle.com), as uploaded by one Mr. Jonathan Besomi. As mentioned by him, the data was collected from [www.superherodb.com](http://www.superherodb.com), & cooked in the nicest & cleanest possible manner in a tabular format.

The dataset named [superheroes\_npl\_dataset.csv](https://drive.google.com/file/d/1h-NCtVhaX80xyyUjxwICKfi3rYzl4sTG/view?usp=sharing) has over 1400+ superheroes from across different superhero universes (Marvel & DC being the major ones). Each row has:

* overall\_score: Derived from [superherodb](http://www.superherodb.com) from the power stats features.
* history\_text: History of the superhero.
* powers\_text: Description of powers.
* intelligence\_score, strength\_score, speed\_score, durability\_score, power\_score, combat\_score: Power stat features.
* full\_name, alter\_ego, aliases, place\_of\_birth, first\_apprearance, base etc.: Origin & connection info.
* gender, type\_race, height, weight, eye\_color etc.: Appearance info.

**Description of Y variable & its type:**

Here, the **overall\_score** will act as our y variable (or the dependent variable). Info regarding **overall\_score** can be seen below:

|  |  |
| --- | --- |
| In: Superheroes[‘overall\_score’].describe() | |
| Out: | |
| count | 1450 |
| unique | 93 |
| top | 6 |
| freq | 162 |
| Name: overall\_score, dtype: object | |

Later, we would be changing the data type to integer for the purpose of building our model.

**Predictor Variables & their meaning:**

Before we get into that, we first extract a subset from **superheroes** using the variable ‘creator’, containing all the DC superhero information & name it **DC\_Comics.**

The shape of **DC\_Comics** is (444, 81).

Looking at all the 81 columns we drop certain columns based on our own understanding of the objective. These columns are:

['real\_name','full\_name','history\_text','powers\_text','place\_of\_birth', 'creator', 'first\_appearance', ‘creator’, 'occupation', 'base', 'relatives', 'type\_race', 'eye\_color', 'hair\_color', 'skin\_color', 'superpowers', 'alter\_egos', 'aliases', 'teams', ‘alignment’, ‘gender’, ‘img’]

We change the variables height & weight (which were initially objects) to integer variables namely ‘height\_cm’ & ‘weight\_kg’ & remove observations which changed from – to NaN in the process. For the scope of this project, we will also remove the categorical variables & focus only on the integer X variables i.e. characteristic scores & y variable i.e. overall\_score. After doing all this & also eliminating multiple rows with numerous null values, now we are left with a dataset of shape (322, 10).

This data set consists of attributes given below:

|  |
| --- |
| <class 'pandas.core.frame.DataFrame'>  Int64Index: 322 entries, 0 to 432  Data columns (total 10 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----  0 name 322 non-null object  1 overall\_score 322 non-null object  2 intelligence\_score 322 non-null int64  3 strength\_score 322 non-null int64  4 speed\_score 322 non-null int64  5 durability\_score 322 non-null int64  6 power\_score 322 non-null int64  7 combat\_score 322 non-null int64  8 height\_cm 322 non-null int64  9 weight\_kg 322 non-null int64  dtypes: int64(8), object(2)  memory usage: 27.7+ KB |

The variables shown here are pretty much self-explanatory.

height\_cm is the height of the superhero, weight\_kg is the weight of the superhero, intelligence\_score is the comparative intelligence the superhero has, strength\_score is the comparative strength the superhero has, same for speed\_score, durability\_score, power\_score & combat\_score, all of which come together & determine the overall\_score of each superhero.

**Preliminary exploratory data analysis:**

The first thing we notice in our y variable while trying to convert it to int64 is that there is at least 1 observation which is ‘∞’ stating that the super hero has an overall\_score of infinity. Even though this can be considered totally justified & true in the super hero world, we would not be able to build our model based on such observations. Outliers will have to dealt with but we start with removing the ‘∞’ variables so that we can change our y variable to int64.

We do this using a while lopp & a count *i*:

|  |
| --- |
| *In:*  i=321  while (i<322) & (i>-1):    i = i-1    if (DC\_Comics2['overall\_score'].iloc[i] == '∞'):      print(i)  *Out:*  *248* |
| *In:*  DC\_Comics2 = DC\_Comics2.drop(index=248)  DC\_Comics2.shape  *Out:*  *(321, 10)* |

Looking as the outputs for .describe(include=[‘int’, ‘object’, ‘category’), we can notice that height\_cm & weight\_kg has some substantial amount of outliers in them. So does strength\_score, power\_score & possibly combat\_score. We confirm this using sns.pairplot():

|  |
| --- |
| *In:*  *sns.pairplot(DC\_Comics2.select\_dtypes(include='int'), size=2)*  *Out:* |

We can see that height\_cm, weight\_kg, overall\_score, intelligence\_score, power\_score & combat\_score have significant outliers. We deal with this using IQR.

The Inter-Quartile Range that we got for our columns is given below:

|  |
| --- |
| *In:*  Q1 = DC\_Comics2.quantile(0.25)  Q3 = DC\_Comics2.quantile(0.75)  IQR = Q3-Q1  print(IQR)  *Out:*  height\_cm 18.0  weight\_kg 30.0  overall\_score 6.0  intelligence\_score 15.0  strength\_score 65.0  speed\_score 45.0  durability\_score 45.0  power\_score 55.0  combat\_score 20.0  dtype: float64 |

Removing the outliers leave us with our final dataset called DC\_Comics\_Clean with no null values or outliers & a shape of (250, 10).

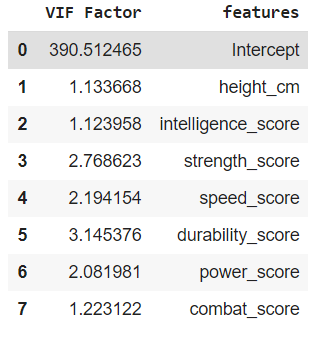
**Significance & effect of Xs on Y:**

To check the effect of our integer Xs we use sns.heatmap() with corr(). What we want is a set of X variables which put substantial effect on our Y variable but not eliminate those which put lesser effect on it. By doing this we will save our model from overfitting.

For this we use the code:

|  |
| --- |
| *In:*  target\_col = "overall\_score"  X = DC\_Comics\_Clean.loc[:, DC\_Comics\_Clean.columns != target\_col]  y = DC\_Comics\_Clean.loc[:, target\_col]  k = 9 #number of variables for heatmap  cols = DC\_Comics\_Clean.corr().nlargest(k, target\_col)[target\_col].index  cm = DC\_Comics\_Clean[cols].corr()  plt.figure(figsize=(14,8))  sns.heatmap(cm, annot=True, cmap = 'viridis', vmin=-1, vmax=1) |

|  |
| --- |
| *Out:* |

 The output shows a good set of co-relations which would work well with our model but we also notice that height\_cm & weight\_kg seem highly co-related. Owing to this we will choose to keep height\_cm & remove weight\_kg because of the minutely higher annotation we got for height\_cm in the heatmap.

We go one step further & generate VIF for the chosen columns. After getting a good enough validation from our VIF scores we head on to perform Multiple Linear Regression on your data set.

**Regression Analysis**

First we start with fitting our X variables & y variable in OLS to estimate the unknown parameters.

The Regression Results we got is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| OLS Regression Results | | | |
| **Dep. Variable:** | overall\_score | **R-squared:** | 0.790 |
| **Model:** | OLS | **Adj. R-squared:** | 0.784 |
| **Method:** | Least Squares | **F-statistic:** | 130.3 |
| **Date:** | Thu, 17 Dec 2020 | **Prob (F-statistic):** | 2.36e-78 |
| **Time:** | 03:11:05 | **Log-Likelihood:** | -529.58 |
| **No. Observations:** | 250 | **AIC:** | 1075. |
| **Df Residuals:** | 242 | **BIC:** | 1103. |
| **Df Model:** | 7 |  |  |
| **Covariance Type:** | nonrobust |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **coef** | **std err** | **t** | **P>|t|** | **[0.025** | **0.975]** |
| **const** | -18.9647 | 2.557 | -7.418 | 0.000 | -24.001 | -13.929 |
| **height\_cm** | 0.0124 | 0.013 | 0.946 | 0.345 | -0.013 | 0.038 |
| **intelligence\_score** | 0.2255 | 0.014 | 15.553 | 0.000 | 0.197 | 0.254 |
| **strength\_score** | 0.0343 | 0.007 | 4.851 | 0.000 | 0.020 | 0.048 |
| **speed\_score** | 0.0404 | 0.007 | 5.393 | 0.000 | 0.026 | 0.055 |
| **durability\_score** | 0.0022 | 0.009 | 0.245 | 0.807 | -0.016 | 0.020 |
| **power\_score** | 0.0585 | 0.007 | 8.707 | 0.000 | 0.045 | 0.072 |
| **combat\_score** | -0.0087 | 0.009 | -1.025 | 0.306 | -0.025 | 0.008 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Omnibus:** | 80.085 | **Durbin-Watson:** | 1.783 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 221.544 |
| **Skew:** | 1.427 | **Prob(JB):** | 7.80e-49 |
| **Kurtosis:** | 6.623 | **Cond. No.** | 4.75e+03 |

Our p-value for height\_cm, durability\_score & combat\_score are significantly higher than 0.05. Thus, we should consider removing these variables. But as we discussed earlier, we do not want to experience overfitting in our model. Thus we select the most in-significant of the 3 i.e. durability\_score & remove that before moving ahead.

We split our data to train & test data (75 – 25 rule) & we train our Linear Regression model on our training set. We then predict on our X\_test & check the regression score for (X\_test, y\_test).

|  |  |
| --- | --- |
| Regression Score | 0.7230 |

Even though the regression score seems good enough & the fit to be perfectly balanced, we perform cross validation to check if our split determined our regression score or not. We also perform Lasso Regression to improve the accuracy of the model.

|  |
| --- |
| Cross Validation Scores (10 folds): |
| array([0.7900, 0.8640, 0.8523, 0.7780 , 0.7921,  0.8887 , 0.7435, 0.5552, 0.4752, 0.8091]) |

Mean: 0.7548

To get the value of alpha for our Lasso we use LassoCV with 5 folds.

|  |  |
| --- | --- |
| *Model\_lassoCV.alpha\_* | 0.0037052067266386394 |

Using this alpha to perform our Lasso Regression.

|  |  |
| --- | --- |
| *Model\_lasso.score(X\_test, y\_test)* | 0.7230 |

Which is the same as our Linear Regression Score proving that our model predictions are at their best accuracy.

**Discussion & Limitations**

The model would work at a 72% accuracy in predicting the Overall Score of a new superhero introduced to it, given the particular characteristic scores are known. This is where the limitation of this model lies. The model will not be able to determine the overall score of an unknown super hero with no background data or scores. If presented with a chance, this project could go further in determining the impact of the 50 categorical variables we had in the original dataset. These variables will not be able to impact this model in any manner because if we assume that 8 of those categorical values, if 1 (yes) lead to an intelligence score of 100, we are basically saying that these 8 variables come together to make the intelligence score on a whole. Including these 8 variables with intelligence score also present will only lead to extremely high multicollinearity in the model. Same goes for all the other characteristic scores & their relation with particular scores.

To deal with this, a set of models will have to be built, each for each scores where the same process will be performed with our y variables respectively being strength\_score, power\_score, durability\_score, speed\_score, intelligence\_score & combat\_score for each of those 6 new models. These 6 models will be working on the concept of predicting the super hero’s individual characteristic scores based only on categorical data (if that super hero has a particular trait or not) & once these 6 scores are generated, they could come together in determining the overall\_score of that new unknown superhero.

**Conclusion**

Nevertheless, this model works well enough in determining the place of each superhero in the DC Comic Universe. Here on we can be certain about who will win in a battle between Flash & Batman, or even Batman (Cristian Bale) & Batman (Ben Affleck).

If by any chance one feels the urge to play Injustice & check the accuracy of this model practically, please record & release the results.

**References**

* Frost, J, Choosing the Correct Type of Regression Analysis, Retrieved from <https://statisticsbyjim.com/regression/choosing-regression-analysis/>
* Besomi, J (Updated: June 2020), Superheroes NLP Dataset, Retrieved from <https://www.kaggle.com/jonathanbesomi/superheroes-nlp-dataset>
* Alexandria, V (August 2020), Superheroes EDA (NLP, Clustering, RF) Python Notebook, Retrieved from <https://www.kaggle.com/alexlichtenberg/superheroes-eda-nlp-clustering-rf>