1. Background

DE2-BIG DATA - Sean B. Lutu, Hannah Qureshi, Aida Manzano Kharman & Felix Murray

1.1 Introduction

The Online News Popularity Dataset is a summary of different attributes of articles published by Mashable.com, a global, multi-platform media and entertainment company. The data was collected over a two year period and has 39797 instances, with 61 attributes.

1.2 Aims

This report sets out to perform predictive analysis on the dataset to establish the link between given attributes of the article and how viral it becomes after being published. By assessing the attributes of the dataset, an initial null hypothesis was proposed based on intuition of what was believed would be the attributes that would influence the article's popularity the most.

This was then compared to the results of the predictive analysis regarding what attributes resulted to be the ones that yielded the highest accuracy model.

1.3 Data Visualisation

In order to understand the unique characteristics of the dataset, which could then be used to train a model in the most effective way, some visualisation was required. A histogram was plotted (Figure 1.3.1) to display the distribution of the data, which made it clear that the dataset was extremely positively skewed. However, in the case of this dataset, the most interesting and useful instances are the outliers, as these are examples of viral behaviour. As a result, the top percentage of the data needed to be captured, rather than the main body.

Another useful visualisation was to produce a scatter plot (Figure 1.3.2). After looking at this, it became clear where useful thresholds could be set to split up the data and distinguish 'viral' articles from 'popular' ones. Using the visualisation and heuristic techniques, it was decided that the threshold for an article being 'viral' would be set at the top 0.1% of the data. However, the dataset was binarised around the 'popular' threshold (top 1%), in order to increase the number of instances available for the training data.

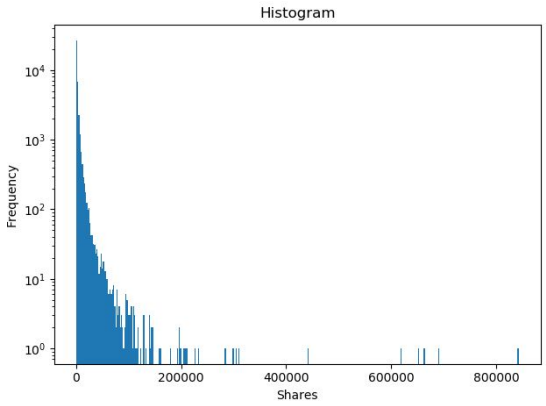


Figure 1.3.1 - Dataset histogram

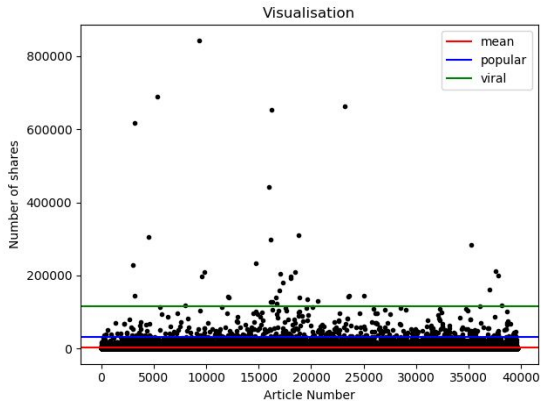


Figure 1.3.2 - Dataset scatter plot

1.4 Data Preparation

Certain attributes were removed; such as the URL attribute, due to it being an object data type. Timedelta (days between the article publication and the dataset acquisition) was also removed as this attribute is unrelated to the article. Finally, the attribute being predicted, the number of shares, was also dropped. The remaining attributes were all numerical data types.

2. Null Hypothesis

Prior to any analysis, it was predicted that the most important attributes that would determine the number of shares of an article would be if its data channel were wither Social Media or Entertainment, the number of words in the title and the number of images in the article. This is because the topics of Social Media and Entertainment have a prevalent online presence, it is assumed that the longer the title the less viral the article will be, and if it has a high number of images it is believed this would make the article more impactful and shareable.

3. Predictive Analysis

3.1 Imbalanced Dataset

Because of the nature of what is being predicted, there are inherently less samples of viral articles than non viral, and as such the dataset needed to be balanced to train the model with an equivalent number of viral and non viral type samples.
Because there were only 38 instances of viral articles, in order to balance our dataset we would have to lose 99% of our data points of non viral article instances. As such the binarising threshold was shifted to include popular as well as viral models in our first bin, and non viral in our second bin. This was a fair compromise between balancing our dataset enough without losing too many sample points.

3.2 Forward and Backwards Selection

Forward and backward selection were used to create a regression model since all attributes were being compared against the binarised number of shares. Shown below are the results from each test.
Forward selection passes:

Attribute Added *(in order)*	kw_max_avg	data_channel_is_world	data_channel_is_entertainment
Accuracy	0.7468	0.8101	0.8228

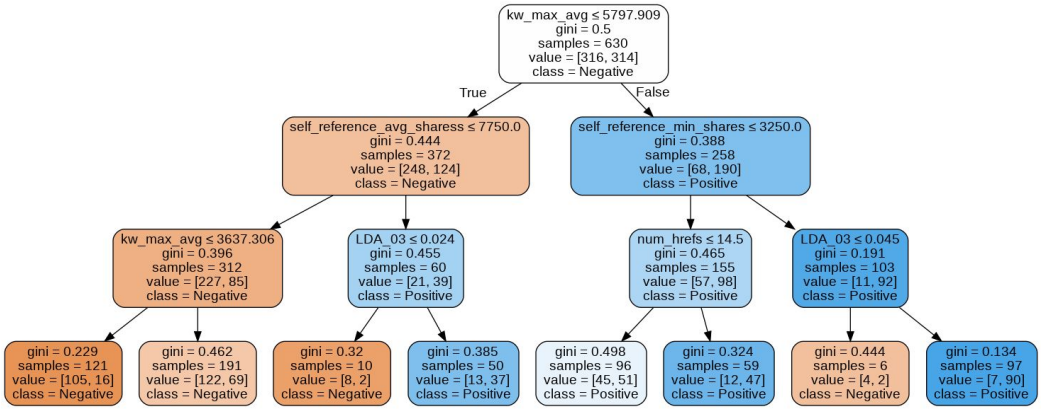
Backward selection passes:

Attribute Removed *(in order)*	kw_min_avg	global_subjectivity	weekday_is_monday	global_sentiment_polarity	kw_avg_min
Accuracy	0.7721	0.7722	0.7595	0.7595	0.7595

As expected, forward selection revealed that the average keyword (max. shares) and world and entertainment data channels had the most significant effect on increasing the models accuracy.

3.4 Decision Tree

It could also be argued that this model is nonlinear, as it is not necessarily a linear function of the variables in it (i.e: an article under the topic of Entertainment is more likely to have higher number of images). As such, the decision tree method to build the predictive model was also used to explore if this could increase the resulting accuracy.



4. Discussion

As expected, forward selection revealed that the average keyword (max. shares) and world and entertainment data channels had the most significant effect on increasing the models accuracy. The accuracy increases from forward selection were; first pass: 0.0633, second pass: 0.0127. This was validated by backward selection since there were no overlapping attributes. A maximum accuracy of 0.8228 was reached.
Using the attributes from forward selection that yielded the greatest accuracy increase in each pass, the decision tree attributes were assessed and the ones not already in the forward selection model that caused the gini to decrease the most were selected.
The final attributes for the model were kw_max_avg, data_channel_is_world, data_channel_is_entertainment, and LDA_03 (closeness of the article topic to the topic no. 3, out of the top 5 most popular topics of the dataset). The ideal outcome was to minimise the number of false negatives since predicting that an article won't go viral (and not publishing it) and it then going viral on a rival platform, would cause a loss in Mashable's platforms exposure. After running the logistic regression model built with the aforementioned attributes, the test set yielded an accuracy of 0.664 (TP:239, FP:110, FN:155, TN:284). Adding more attributed to the model resulted in an accuracy drop to 0.53. We are therefore confident in the effectiveness of the model.

6 Appendix:

6.1 Data Source

http://archive.ics.uci.edu/ml/datasets/Online+News+Popularity# (contains a description of all attributes for clarity)

6.2 Loading the dataset

```
# load OnlineNewsPopularity file
unsampled_news_pop = pd.read_csv('OnlineNewsPopularityProcessed.csv')
# print(unsampled_news_pop)

# strip out space before column names
unsampled_news_pop.columns = [i.strip() for i in unsampled_news_pop.columns]
# print(list(unsampled_news_pop.columns))
```

6.3 Visualisation

```
# plot histogram
bin_num = int(len(unsampled_news_pop)/100)
plt.hist(unsampled_news_pop['shares'], bins=bin_num, log=True)
plt.title("Histogram")
plt.ylabel('Frequency')
plt.xlabel('Shares')
plt.tight_layout()
plt.show()

# find mean number of shares
shares_mean = unsampled_news_pop.loc[:,'shares'].mean() # 3395
# print(shares_mean)

# find top 1% of number of shares - these articles are 'popular'
one_pct = int(len(unsampled_news_pop)*0.01)
top_one_pct = unsampled_news_pop.nlargest(one_pct, 'shares')
# print(top_one_pct['shares'])
popular_thresh = top_one_pct['shares'].min() # 31900

# find top 0.1% of number of shares - these articles are 'viral'
pt_one_pct = int(len(unsampled_news_pop)*0.001)
top_pt_one_pct = unsampled_news_pop.nlargest(pt_one_pct, 'shares')
# print(top_pt_one_pct['shares'])
viral_thresh = top_pt_one_pct['shares'].min() # 115700

# visualise data
news_length = list(range(0, len(unsampled_news_pop)))
plt.scatter(news_length, unsampled_news_pop['shares'], c='k', marker='.')

# plot threshold lines
plt.axhline(y=shares_mean, color='r', linestyle='-', label='mean')
plt.axhline(y=popular_thresh, color='b', linestyle='-', label='popular')
plt.axhline(y=viral_thresh, color='g', linestyle='-', label='viral')

# plot labels
plt.title("Visualisation")
plt.xlabel('Article Number')
plt.ylabel('Number of shares')
plt.legend()
plt.tight_layout()
plt.show()
```

6.4 Preparation

```
# check data types to determine which columns to remove immediately
print(unsampled_news_pop.dtypes)

# binarise 'shares' in new column 'shares_bin' (around 'popular' threshold)
unsampled_news_pop['shares_bin'] = (unsampled_news_pop['shares'] > popular_thresh).astype(int)
# print(news_pop['shares_bin'])

# check to make sure data has binarised correctly
bin_test = int(len(unsampled_news_pop)*0.002)
bin_testing = unsampled_news_pop.nlargest(bin_test, 'shares')
# print(bin_testing)

# undersample majority class
total = len(unsampled_news_pop)
nb_popular = unsampled_news_pop['shares_bin'].sum()
nb_non = total - nb_popular
news_pop_popular = unsampled_news_pop.loc[unsampled_news_pop['shares_bin'] == 1]
news_pop_non = unsampled_news_pop.loc[unsampled_news_pop['shares_bin'] == 0].sample(nb_popular)
news_pop = pd.concat((news_pop_popular, news_pop_non))
# these numbers should be the same
print('The sizes of the minority (popular) class and majority (non-viral) class are {}'.format((len(news_pop_popular), len(news_pop_non))))

# drop object column (url) and python timing column (timedelta)
news_pop = news_pop.drop(['url', 'timedelta'], axis=1)

# now remove 'shares' as this is what we test for
news_pop = news_pop.drop(['shares'], axis=1)
# print(news_pop)

# split into train,validation and test sets
train, other = train_test_split(news_pop, test_size=0.2, random_state=0)
validation, test = train_test_split(other, test_size=0.5, random_state=0)
print('The sizes for train, validation and test should be {}'.format((len(train), len(validation), len(test))))
```

6.7 Forward & Backward Selection

```
class Finished(Exception):pass

# split sets into x and y folds
X_train = train.drop(columns=['shares_bin'])
y_train = train['shares_bin']

X_val = validation.drop(columns=['shares_bin'])
y_val = validation['shares_bin']

X_test = test.drop(columns=['shares_bin'])
y_test = test['shares_bin']

# automatic forward selection
columns_to_test = X_train.columns
columns_in_model = []
for i in range(0,60):
    columns_in_model_updated, acc_best = func.select_column_to_add(X_train, y_train, X_val, y_val, columns_in_model, columns_to_test)
    columns_in_model = columns_in_model_updated

# automatic backward selection
columns_to_test = X_train.columns
columns_in_model = columns_to_test
for i in range(0,60):
    columns_in_model_updated, acc_best, columns_in_model = func.select_column_to_remove(X_train, y_train, X_val, y_val, columns_in_model, columns_to_test)
    columns_to_test = columns_in_model_updated
    columns_in_model = columns_in_model_updated
```

6.8 Decision tree minimum impurity decrease (per each iteration)

```
dt = DecisionTreeClassifier(max_depth = 3)
dt = dt.fit(X_train, y_train)

dot_data = tree.export_graphviz(dt, out_file=None)
graph = graphviz.Source(dot_data)

predictors = X_train.columns
#print(predictors)
dot_data = tree.export_graphviz(dt, out_file=None,
                                feature_names = predictors,
                                class_names = ('Negative', 'Positive'),
                                filled = True, rounded = True,
                                special_characters = True)
graph = graphviz.Source(dot_data)
graph
```

6.9 Logistic Regression Model (testing on test set)

```
selected_attributes = ['kw_max_avg', 'data_channel_is_world', 'data_channel_is_entertainment', 'LDA_03']
X = news_pop[selected_attributes]
y = news_pop['shares_bin'].values.reshape(-1,1)

mylr = logreg()
mylr.fit(X, y)

model_summary = func.ModelSummary(mylr, X, y)
model_summary.get_summary()
```

```
The sizes of the minority (popular) class and majority (non-viral) class are (394, 394)
The sizes for train, validation and test should be (630, 79, 79)
                              coefficient    std  p-value  [0.025  0.975]
intercept                          -0.831  0.174    0.000  -1.173  -0.490
kw_max_avg                          0.000  0.000    0.003   0.000   0.000
data_channel_is_world              -0.390  0.222    0.079  -0.826   0.046
data_channel_is_entertainment      -0.313  0.201    0.119  -0.707   0.081
LDA_03                              1.587  0.254    0.000   1.089   2.085
------------------------------------------------------------
Confusion Matrix (total:788)     Accuracy:        0.664
  TP: 239 | FN: 155
  FP: 110 | TN: 284
[Finished in 6.172s]
```