Predicting viewership fall-off for an advertisement based on network and show

Rishi Dabre
College of Electrical Engineering and
Computer Science
Syracuse University
Syracuse, NY 13244
Email: rrdabre@syr.edu

***Abstract*—Often at an interesting moment from our favorite program, it pauses for a commercial advertisement. Adding to the annoyance is the fact that the advertisement appears completely out of place to us, in that situation comprising of factors like time of the day, type of the program, geographical location and at times, even our age. Situations like this, we believe, cause a significant viewership fall-off all over the television network. Our project aims at predicting to what extent, a particular brand of advertisements is likely to cause a fall-off in the number of viewers given the network is it being aired on and the show it is placed with. Our results indicate that this prediction cannot be based solely on these two parameters and and requires more detailed research to help television networks and brands collectively choose the timings, audiences and so on for their programs and advertisements more effectively.**

***Keywords*—advertisement, viewership, network, show.**

I. Introduction

Our aim is to be able to predict the extent to which a particular brand of advertisements is likely to expect a fall-off in the number of viewers given the network it is being aired on and the show it is placed with. Provided the results are promising, this analysis could help the television networks and the brands cut down on the huge cost of finding such viewership results out empirically.

The basic idea is to use as minimal yet influential attributes as possible in predicting the changes in the viewership. Our intuition is that the network and the particular show the advertisements are aired with are just such factors. We further qualify our statement adding that these are not exhaustive and there might be other more influential factors. However, our goal for this project is to determine, in the worst case, to what extent do these factors impact the viewership fall-off.

II. Problem & Data Description

Currently, ComScore[1] has no such data available that talks about the factors affecting viewership of the broadcasts. They however collect all the statistics about them all and make everything available very conveniently through Rentrak[2]. A great fit for such a problem comprising of largely available data set with a significant expectation of predictability due to human involvement is therefore an Artificial Neural Network.

We obtained the advertisement data in the form of reports generated from the Rentrak website[3]. These reports consist of columns stating the ad occurrence information including but not limited to show name, television network name, average number of audience tuned into the television, average number of audience watching the advertisement, demographic statistics, advertisement pod info and even ad length. For the project however, we are concerned with only the following fields:

1) **Tlcst** which is the name of the show the ad is being aired with.
2) **Net** which is the name of the network the ad is being aired on.
3) **Average Telecast Audience (Avg Tlcst Aud)** which is the average number of TVs tuned into the telecast throughout its run time (calculated by dividing the total hours viewed by the run time in hours).
4) **Average Ad Copy Audience (Avg Ad Copy Aud)** which is the average number of TVs tuned into the ad throughout its duration (calculated by dividing the total seconds the ad was viewed by the duration of the ad in seconds).

III. Approach

On a high level, our approach consisted of the following two steps:

A. Building the input data set

We collected the advertisement data for roughly the last whole year. As multiple advertisements (of a specific brand) occurred in this period, we began by selecting the ones aired for the maximum total time in each case. Next, we manually extracted the required columns, merged all the reports and passed them to a python script in order to obtain the consolidated data set file containing integer coded attributes.

The output field, viewership fall-off was not directly available in the reports which was precisely the reason why we retained the columns talking about the number of audience. In order to obtain values for the viewership fall-off (f) column, we used the following formula:

$$f = (t - a)/a \tag{1}$$

Here, t is the Average Telecast Audience (Avg Tlcst Aud) and a is the Average Ad Copy Audience (Avg Ad Copy Aud). However, when there was no fall-off or instead increase instead in

viewership, this value was treated zero (as a Winsorizing step) and not negative. The network and show names, which were strings, were then translated into inputs for the neural network by the script encode.py. It took as an argument the consolidated input file, assigned a number to each network (remembered in network_map.txt) and show (show_map.txt) and wrote the encoded entries in a new csv file (data_consolidated.txt), ready to be supplied to the neural network.
Following is the description of the data set thus formed:

- Number of attributes: 2
- Number of instances: 2553
- Attributes: Network ID and Show ID (integers)
- Output: Viewership fall-off (reals)

B. Developing and training the network

We used multiple different implementations of the Backpropagation algorithm[4][5][6] for predicting the viewership fall-off. With each one, we trained the network on the exhaustive data set formed, used the same to predict the fall-off for the very same data and analyzed the results obtained.

C. Algorithm

We began our initial experiment with the detailed python implementation of Backpropagation algorithm[4] available at Machine Learning Mastery by Jason Brownlee, an experienced Machine Learning professional. Using a single hidden layer with 2-10 nodes, this implementation yielded drastically bad results irrespective of whether it was trained on a subset of the data set or the whole of it. Due to this, we switched to another Backpropagation implementation. This one was a simple 3 layer python implementation by I am trask[5], a PhD student at the University of Oxford. We extended this code and tailored it to our particular data set and to plot the output values against the expected ones using the python matplotlib library[7] provided by SciPy[8].

In general, all of our algorithm implementations took the input file name in an argument, constructed an in memory data set from the file and normalized the same following the feature scaling technique[9]. Further, it used the sigmoid function for the initial assignment of weights to the nodes and gradient descent to find the error at both the hidden and the output layers. The weights were then trained for a total of 100,000 iterations followed by computing the error corresponding to each input row. Finally, error values and output values against expected ones were plotted.

Reading from the results of the experiments so far, we performed a series of iterations as a part of our regression analysis as follows:

1) Iteration 1: Changing the algorithm and comparing with Iris: The results obtained so far were not up to the expectation. Our algorithm did not do well in learning and predicting the data set outputs. In order to make sure this particular implementation of the algorithm had no detrimental impact on the unwarranted results, we switched to another implementation of Backpropagation algorithm by professors Li, Johnson and Yeung from the Stanford University[6]. As another important guideline to test the algorithm on different data sets, we used the same algorithm on the iris data set from the UCI Machine Learning repository[10]. The specifications of this algorithm were as follows:

- Number of layers: 3
- Number of nodes in hidden layer: 3
- Learning rate: 1
- Number of iterations: 20000

We ran the algorithm on both our original and the iris data sets. This time, we used the same data set each time for both training and testing the neural network.

D. Iteration 2: Trimming the Data Set

As the behavior of the Rentrak data set still appeared confounding to the algorithm, we decided to conduct the experiment with some more changes. Using the same network, the reliability of which had been established in the last iteration, we ran some more tests in the manner described next.

Our intuition was that the data set was likely to follow the long tail distribution. In precise words, we expected the data set to contain a little too many instances where the fall-off in viewership happened from a very small actual number in the orders of 10s to 100s of thousands. These instances, we believed, interfered with the learning process of our neural network. Therefore, we decided to cut this long tail off by taking off from the data set, the instances where the fall-off happened from a number below 500,000. Our expectations were that this trimming of data set would render itself predictable while at the same time convenient to conduct experiments extensively with.

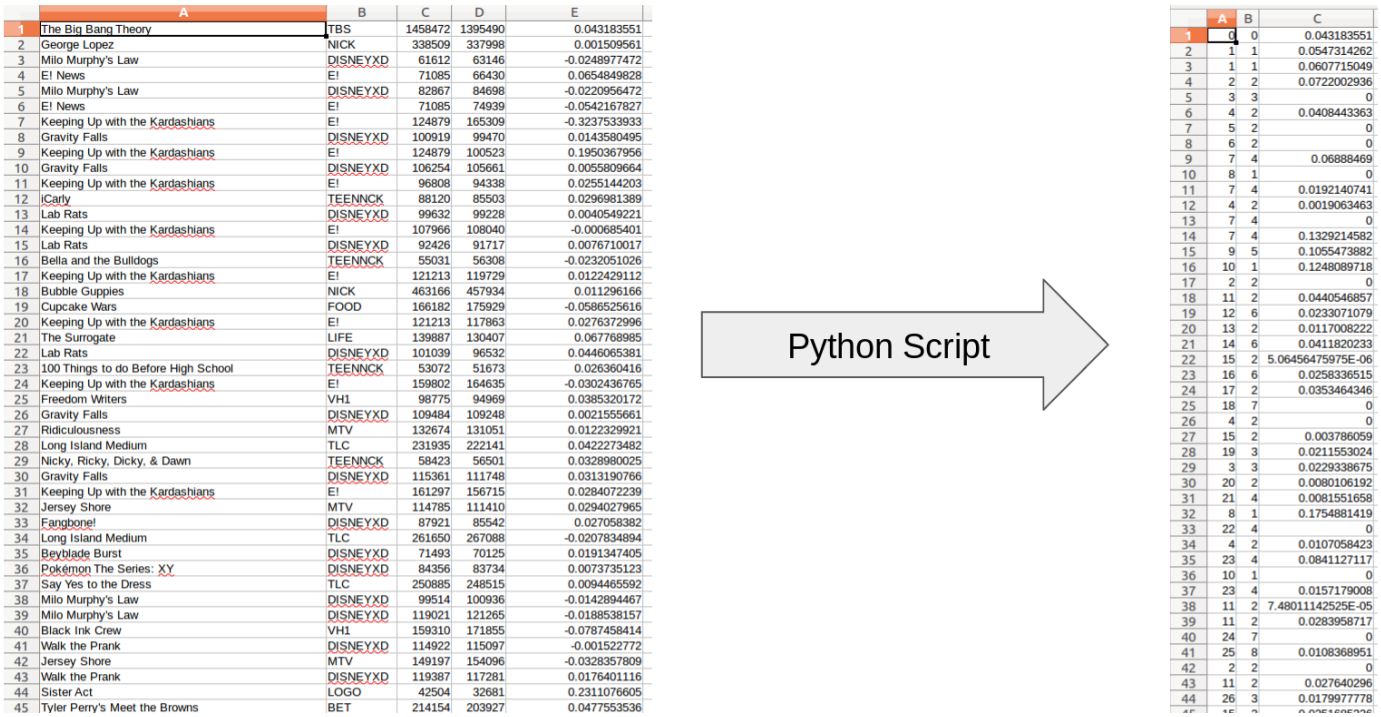
Furthermore, looking at the previous results wherein the predictions spanned too small a range of values, we adjusted the initial weights of the nodes from the first and the hidden layers so as to kick off the predicted values right from the beginning. Additionally, we increased the number of learning iterations from 200,000 to 1,500,000 and reduced the number of hidden nodes from 3 to 2 (as it appeared to reduce the mean squared error quicker). Finally, we changed the backpropagation function from the one simply amplifying the output difference to the first derivative or the gradient descent function.

E. Iteration 3: Clubbing the Inputs

Concluding from the observations of the above visualization, we decided to manipulate the input fields a little. To be specific, we clubbed the network and show names together to form a single string which we then indexed into an array using another python script. This way, we obtained a single input column for a corresponding output column and ran the same neural network on this data set.

IV. Data

The data from reports was translated to the appropriate input format as show in figure 1. We plotted all the columns of our data set as supplied to the neural network (i. e. obtained by processing the reports and normalizing the values) looked



	A	B	C	D	E
1	The Big Bang Theory	TBS	1458472	1395490	0.043183551
2	George Lopez	NICK	338509	337998	0.001509561
3	Milo Murphy's Law	DISNEYXD	61612	63146	-0.0248977472
4	E! News	E!	71085	66430	0.0654849828
5	Milo Murphy's Law	DISNEYXD	82867	84698	-0.0220956472
6	E! News	E!	71085	74939	-0.0542167827
7	Keeping Up with the Kardashians	E!	124879	165309	-0.3237533933
8	Gravity Falls	DISNEYXD	100919	99470	0.0143580495
9	Keeping Up with the Kardashians	E!	124879	100523	0.1950367956
10	Gravity Falls	DISNEYXD	106254	105661	0.0055809664
11	Keeping Up with the Kardashians	E!	96808	94338	0.0255144203
12	iCarly	TEENNICK	88120	85503	0.0296981389
13	Lab Rats	DISNEYXD	99632	99228	0.0040549221
14	Keeping Up with the Kardashians	E!	107966	108040	-0.000685401
15	Lab Rats	DISNEYXD	92426	91717	0.0076710017
16	Bella and the Bulldogs	TEENNICK	55031	56308	-0.0232051026
17	Keeping Up with the Kardashians	E!	121213	119729	0.0122429112
18	Bubble Guppies	NICK	463166	457934	0.011296166
19	Cupcake Wars	FOOD	166182	175929	-0.0586525616
20	Keeping Up with the Kardashians	E!	121213	117863	0.0276372996
21	The Surrogate	LIFE	139887	130407	0.067768985
22	Lab Rats	DISNEYXD	101039	96532	0.0446065381
23	100 Things to do Before High School	TEENNICK	53072	51673	0.026360416
24	Keeping Up with the Kardashians	E!	159802	164635	-0.0302436765
25	Freedom Writers	VH1	98775	94969	0.0385320172
26	Gravity Falls	DISNEYXD	109484	109248	0.0021555661
27	Ridiculousness	MTV	132674	131051	0.0122329921
28	Long Island Medium	TLC	231935	222141	0.0422273482
29	Nicky, Ricky, Dicky, & Dawn	TEENNICK	58423	56501	0.0328980025
30	Gravity Falls	DISNEYXD	115361	111748	0.0313190766
31	Keeping Up with the Kardashians	E!	161297	156715	0.0284072239
32	Jersey Shore	MTV	114785	111410	0.0294027965
33	Fangbone!	DISNEYXD	87921	85542	0.027058382
34	Long Island Medium	TLC	261650	267088	-0.0207834894
35	Beyblade Burst	DISNEYXD	71493	70125	0.0191347405
36	Pokémon The Series: XY	DISNEYXD	84356	83734	0.0073735123
37	Say Yes to the Dress	TLC	250885	248515	0.0094465592
38	Milo Murphy's Law	DISNEYXD	99514	100936	-0.0142894467
39	Milo Murphy's Law	DISNEYXD	119021	121265	-0.0188538157
40	Black Ink Crew	VH1	159310	171855	-0.0787458414
41	Walk the Prank	DISNEYXD	114922	115097	-0.001522772
42	Jersey Shore	MTV	149197	154096	-0.0328357809
43	Walk the Prank	DISNEYXD	119387	117281	0.0176401116
44	Sister Act	LOGO	42504	32681	0.2311076605
45	Tyler Perry's Meet the Browns	BET	214154	203927	0.0477553536

	A	B	C
1	0	0	0.043183551
2	1	1	0.0547314262
3	1	1	0.0607715049
4	2	2	0.0722002936
5	3	3	0
6	4	2	0.0408443363
7	5	2	0
8	6	2	0
9	7	4	0.06888469
10	8	1	0
11	7	4	0.0192140741
12	4	2	0.0019063463
13	7	4	0
14	7	4	0.1329214582
15	9	5	0.1055473882
16	10	1	0.1248089718
17	2	2	0
18	11	2	0.0440546857
19	12	6	0.0233071079
20	13	2	0.0117008222
21	14	6	0.0411820233
22	15	2	5.06456475975E-06
23	16	6	0.0258336515
24	17	2	0.0353464346
25	18	7	0
26	4	2	0
27	15	2	0.003786059
28	19	3	0.0211553024
29	3	3	0.0229338675
30	20	2	0.0080106192
31	21	4	0.0081551658
32	8	1	0.1754881419
33	22	4	0
34	4	2	0.0107058423
35	23	4	0.0841127117
36	10	1	0
37	23	4	0.0157179008
38	11	2	7.48011142525E-05
39	11	2	0.0283958717
40	24	7	0
41	25	8	0.0108368951
42	2	2	0
43	11	2	0.027640296
44	26	3	0.0179977778

Fig. 1: Translating data from raw reports into input attributes

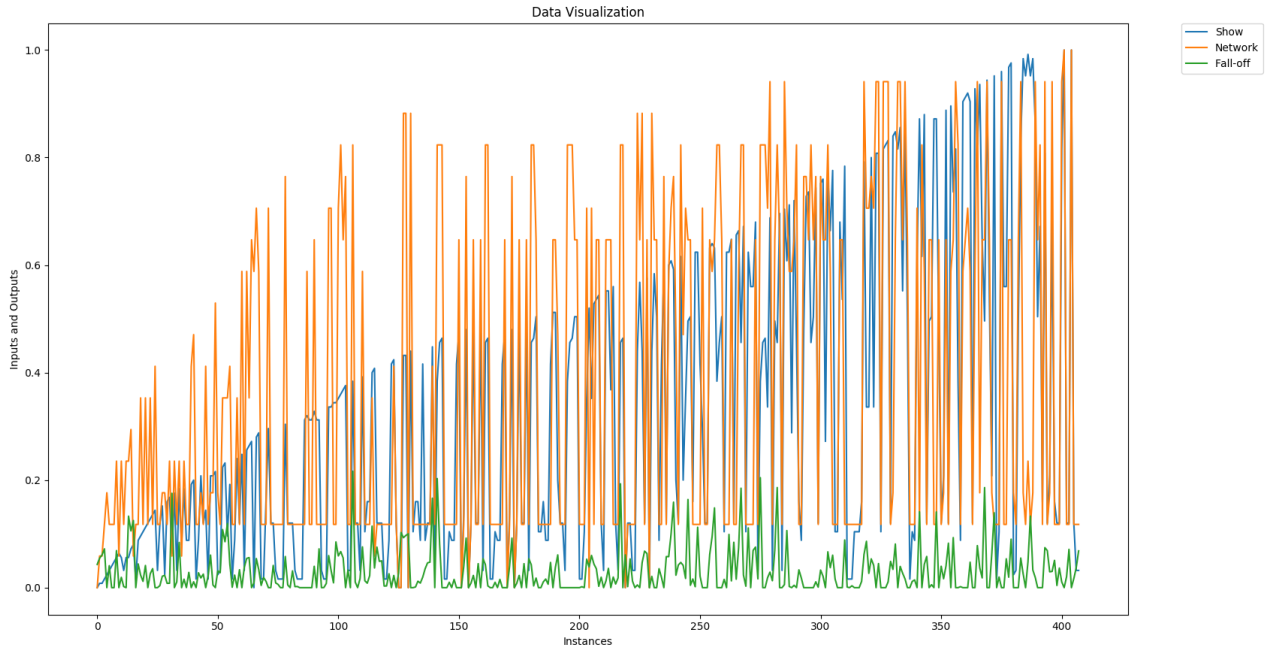


Fig. 2: Visualization of original data set (network, show and fall-off values)

as shown in figure 2. The plot of the data set, after clubbing the input fields network and show to form a unique entity looked as shown in figure 3. From the plot in figure 3, it could be very well inferred that the output values did not necessarily show any strong correlation with any of the input column values.

V. Results

Figure 4 shows the error (along the y-axis) appeared in the actual output values compared to the expected output values for all the instances (along the x-axis). Figure 5 shows the actual values against the expected values of the output. From 4 and 5, it can be observed that approximately the latter half of data set behaved less predictably according to the algorithm. Roughly, the errors appeared to be only increasing with the number of instances looked at.

A. Iteration 1

Looking at figures 6, 7, 8 and 9, it appears that the outputs for the iris data set were predicted relatively better than those for the Rentrak data set. To be more precise, the obtained output values in case of the iris data set were much closer to the expected ones while those in case of the Rentrak one remained strictly within the interval (0.00, 0.05). An advantage that the iris data set had over the Rentrak one was that it merely consisted of three possible output values for each row instance. But Rentrak had different output values for all the input rows. While this appeared as a potentially helping factor in the better predictability of the iris data set, it was limited by the number of instances iris comprised of which was way lesser than that in Rentrak consisting of around 2500 instances that could help the neural network learn better due to such large number of instances. Yet instead of expecting reliability due to use of the complete data set for training, we used a subset of around 500 instances for the Rentrak data set and then tested the network over the whole data set. The results still appeared to be extremely similar to the ones observed before.

B. Iteration 2

The mean squared error appeared to drop no lower than approximately 0.0016. The errors and the results are shown in figures 10 and 11. As can be observed from 10 and 11, reducing the number of data set instances from a total of 2500 to around 400 by cutting off the apparently insignificant rows too did not help improve the results. While the prediction range this time widened by 0.05 both above and below, that seemed only an effect of the readjusted initial weights. Accounting for the possibility that the learning did improve, when we ran the network for even higher number of iterations (around 2,000,000), the mean squared error always seemed to stabilize after approximately 1,500,000.

C. Iteration 3

Clubbing the columns did not seem to work either as is clearly visible in figure 12 and figure 13. Even this data set behaved almost the same as the original one. Moreover, the clubbing process appeared to have created the distribution of the data set just similar to that seen before. It can also be distinctly noticed that the major influence on the clubbing was that of the shows than that of the networks, which was obvious given the extremely small number of networks involved as compared to that of shows.

VI. Discussion

After an exhaustive analysis of the Rentrak data set using different algorithms configured at various parameter values, it can be safely assumed that the Rentrak data set, in particular the one we formed, behaved unpredictably. We strongly feel that the data translation of the original data set obtained from the Rentrak reports could have been carried out in a more significant manner and the results could have been different.

Furthermore, our data set comprised of the advertisement occurrences spanning across a whole year for all the timings which also seems to have added to the confusion in the learning of neural network obstructing it from molding itself to be able to better predict the values for the very inputs it learned in the beginning. Moreover, we expect other factors like the actual time or slot (prime, noon, etc.) of airing, day of the week and more such significant ones to affect the viewership. Thus, the viewership fall-off for the advertisements of the Target brand and Apparels category cannot be predicted accurately based solely on the network and the show they are aired with.

Acknowledgment

I would like to thank Prof. Mohan for the invaluable guidance he has given me throughout the project. It would have been very difficult had it not been for his support. Also, I would like to express sincere gratitude towards Prof. Egan and Prof. Chew for their belief in us and motivating us right from the beginning. Finally, I would like to thank my colleagues Kartik and Sushanth for being with me and giving their time for discussions.

References

[1] ComScore
https://www.comscore.com/

[2] Rentrak by ComScore
https://national-tv.rentrak.com

[3] Ad Occurrence Search
https://national-tv.rentrak.com/reports/ad_occurrence.html

[4] How to Implement the Backpropagation Algorithm From Scratch In Python
https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-p

[5] A Neural Network in 11 lines of Python (Part 1)
https://iamtrask.github.io/2015/07/12/basic-python-network/

[6] Lecture 4: Backpropagation and Neural Networks
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture4.pdf

[7] Matplotlib: Python plotting – Matplotlib 2.1.0 documentation
https://matplotlib.org/users/pyplot_tutorial.html

[8] SciPy.org
https://scipy.org/

[9] Feature scaling
https://en.wikipedia.org/wiki/Feature_scaling

[10] UCI Machine Learning Repository: Iris Data Set
https://archive.ics.uci.edu/ml/datasets/iris

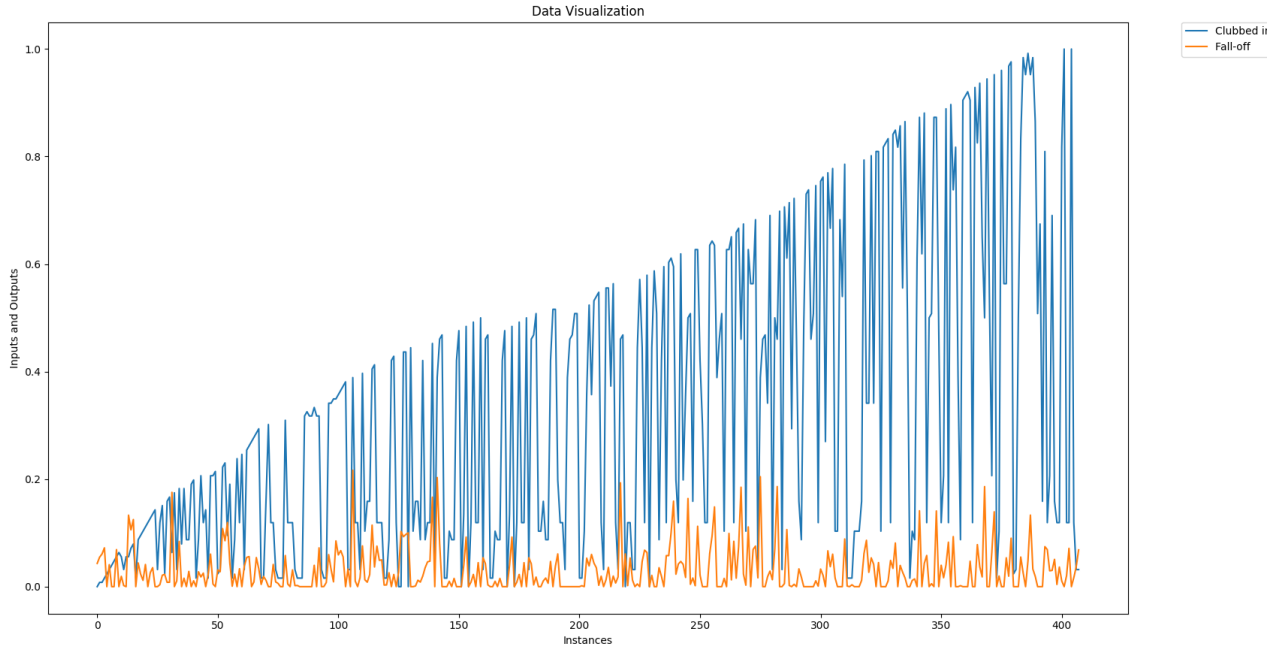


Fig. 3: Visualization of data set after clubbing network and show

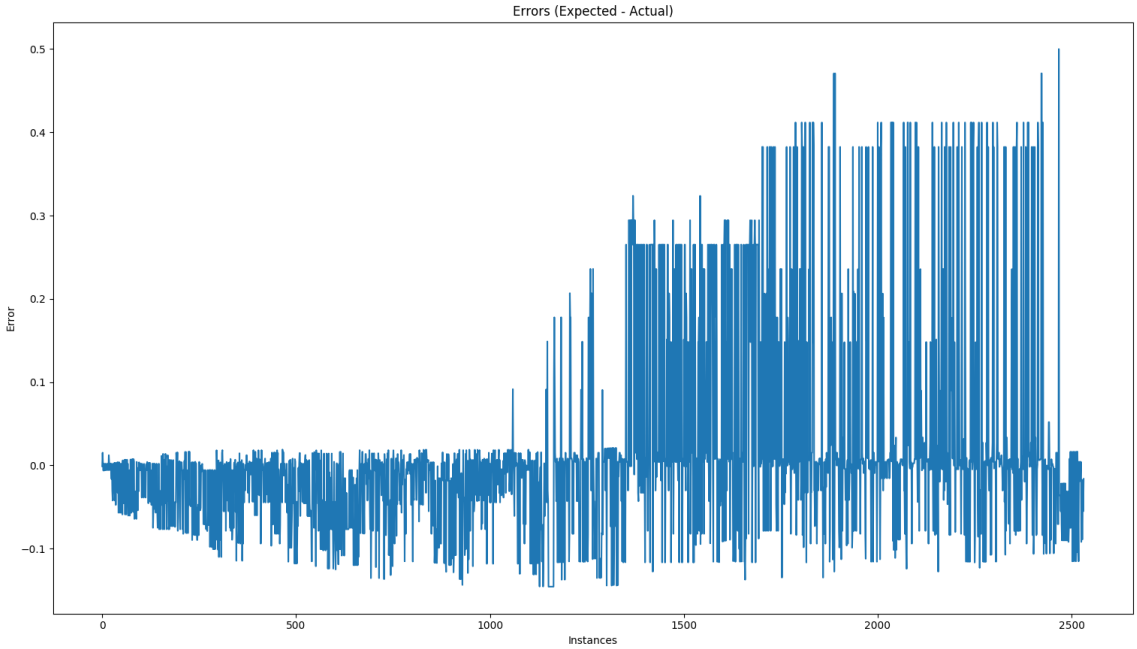


Fig. 4: Error values against instance numbers

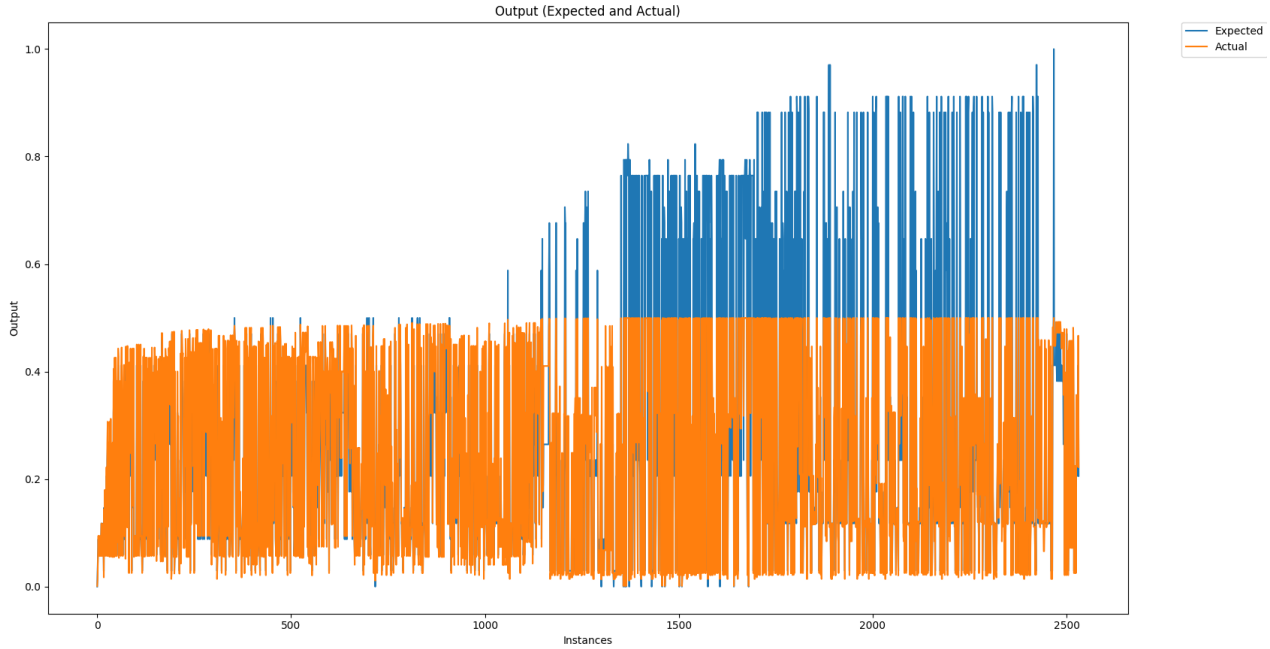


Fig. 5: Expected and actual output values against instance numbers

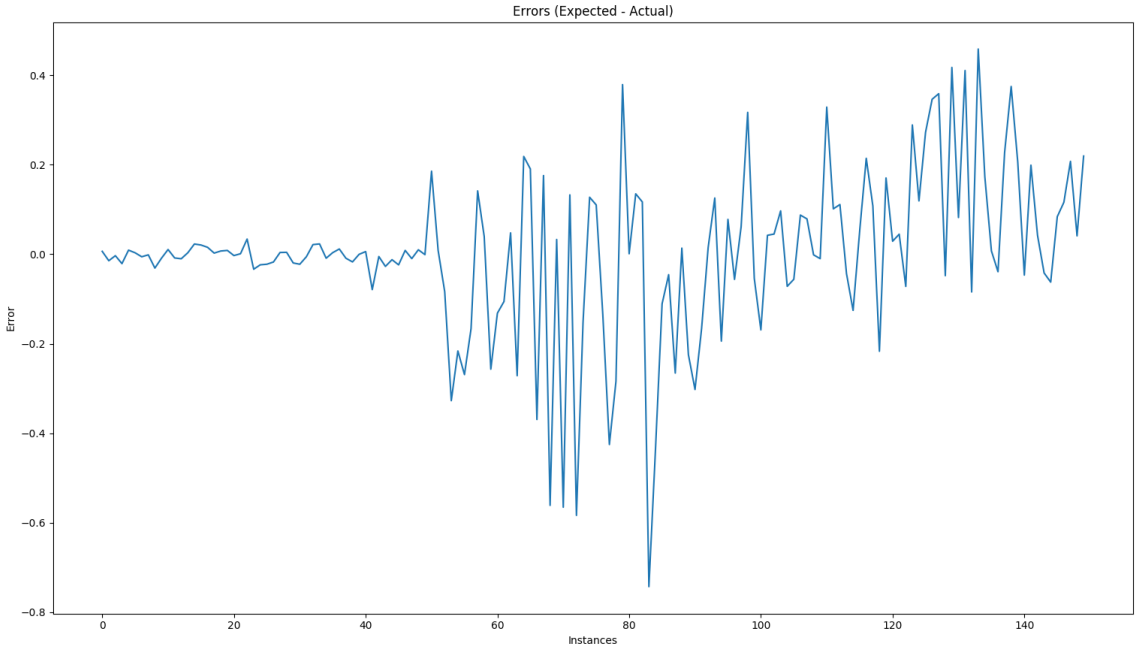


Fig. 6: Error values against instance numbers

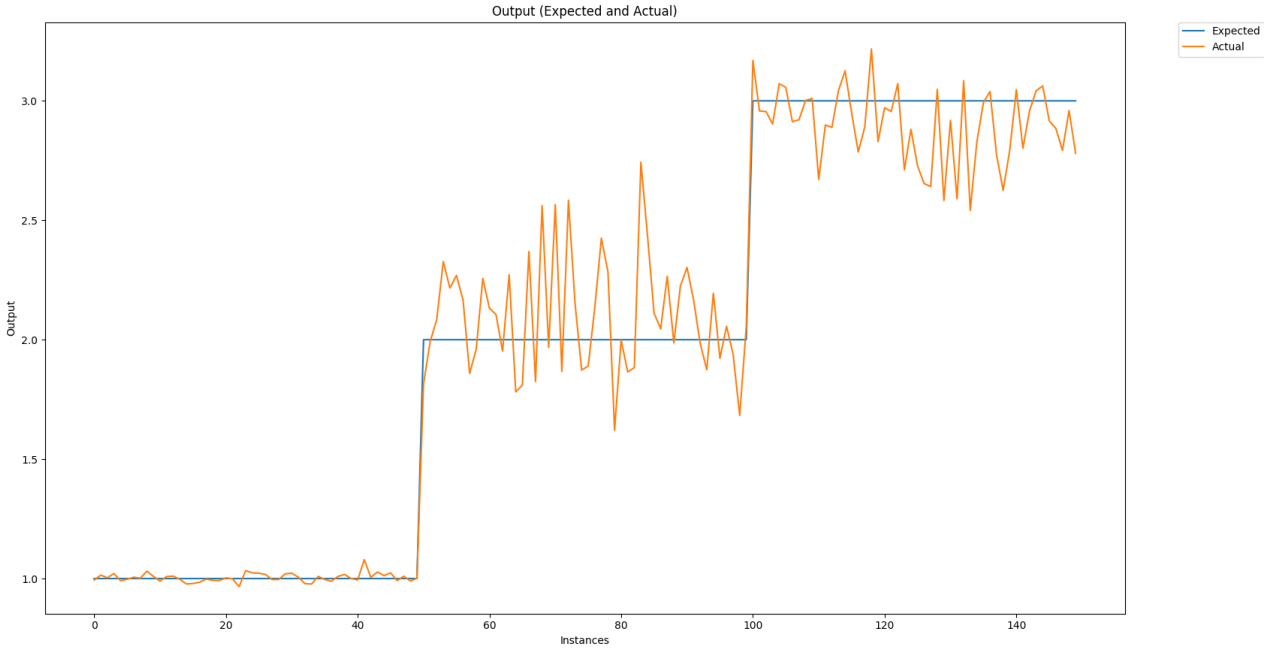


Fig. 7: Expected and actual output values against instance numbers

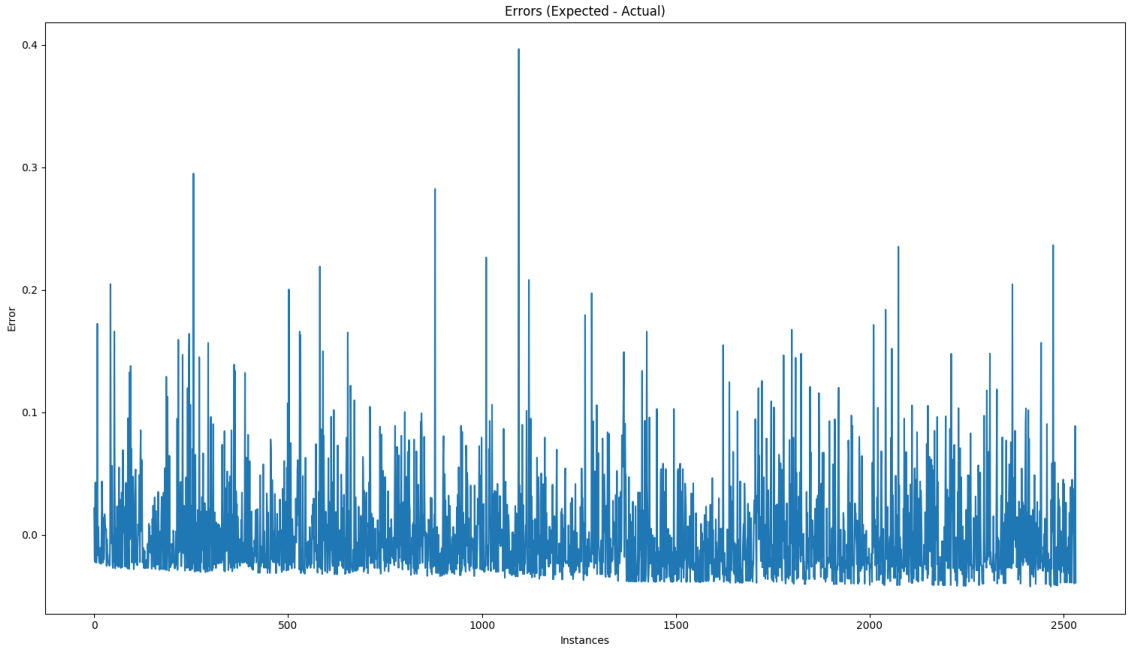


Fig. 8: Error values against instance numbers

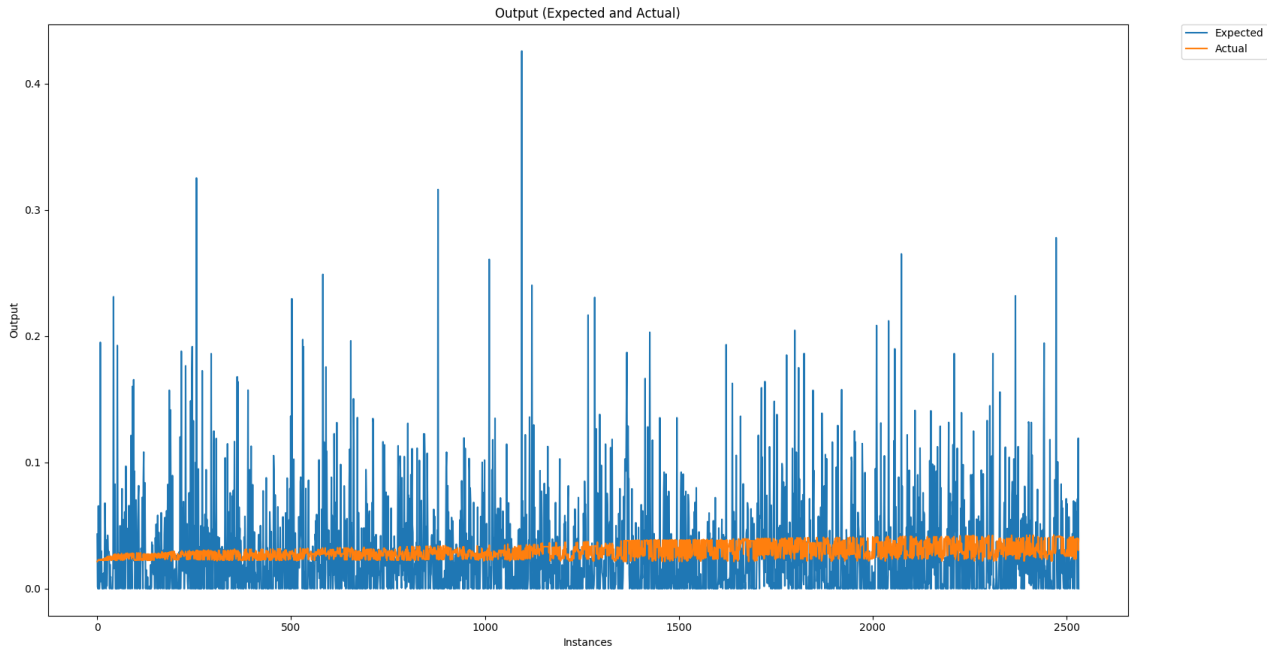


Fig. 9: Expected and actual output values against instance numbers

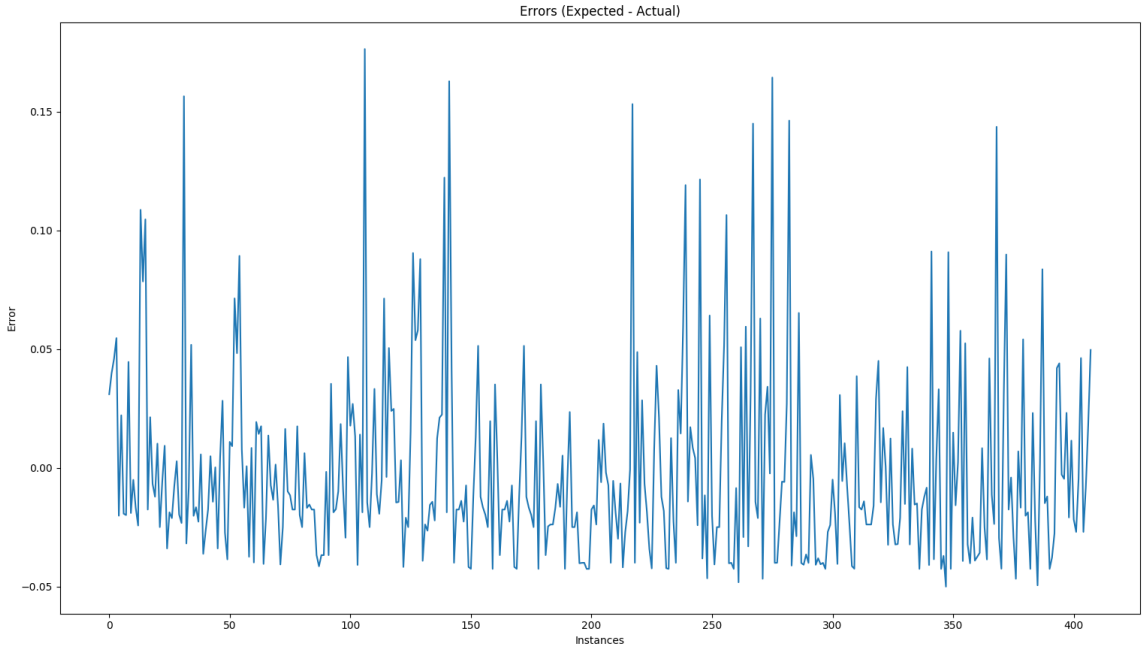


Fig. 10: Error values against instance numbers

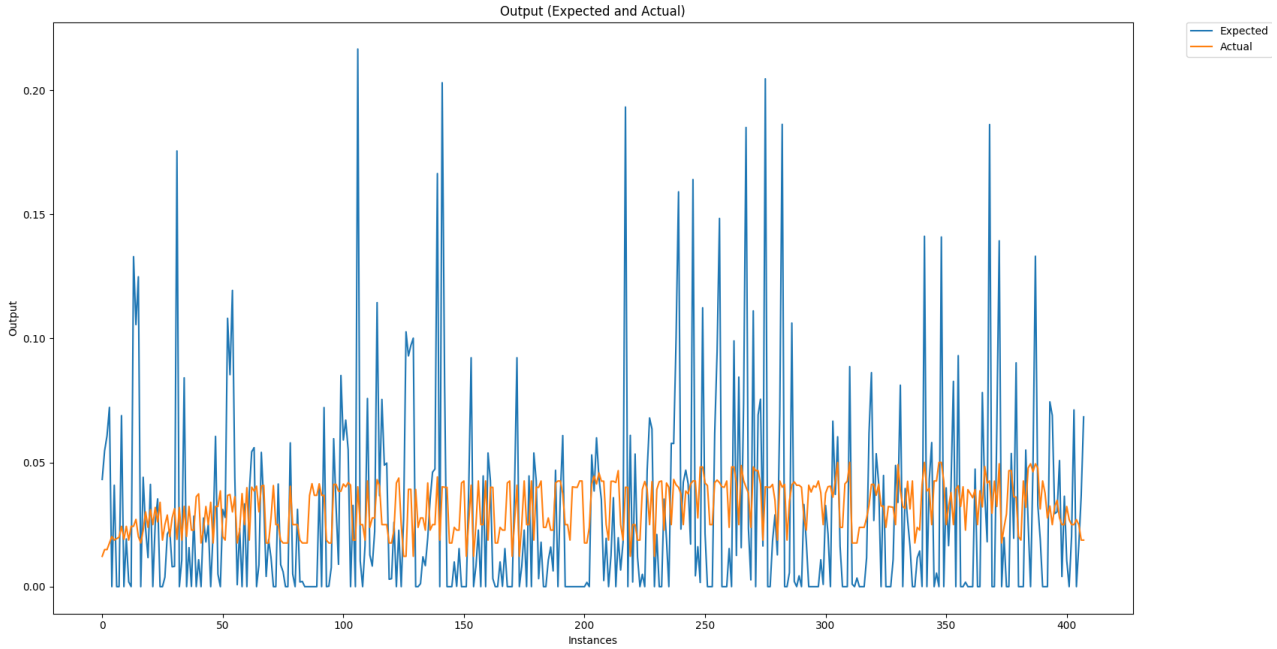


Fig. 11: Expected and actual output values against instance numbers

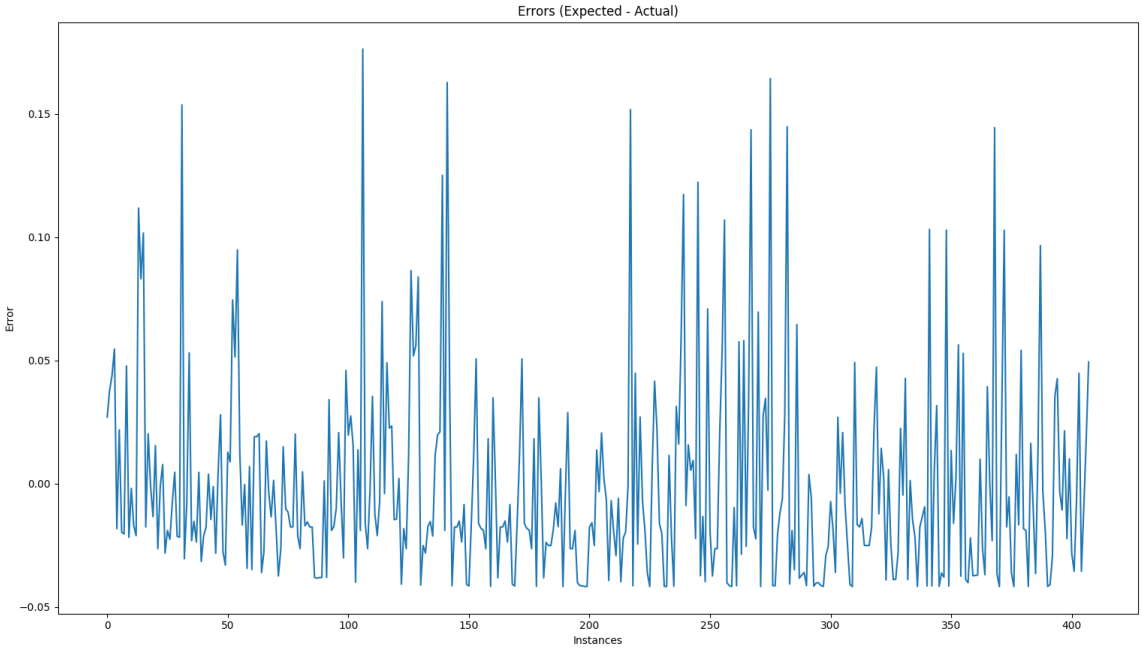


Fig. 12: Error values against instance numbers

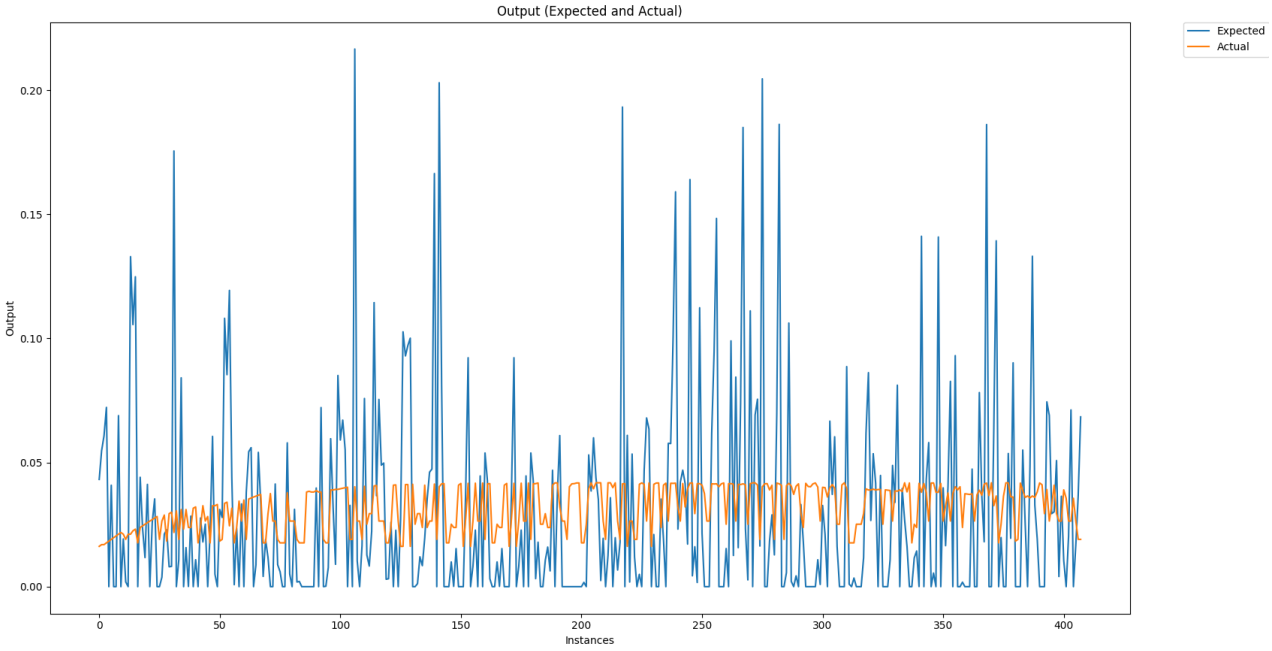


Fig. 13: Expected and actual output values against instance numbers