# Summary of dataset

|  |  |
| --- | --- |
| Predictor variables | 53 |
| Dependent variable | No. of comments in next H hours(output) |
| Number of observations | 82312 |
| Missing values | None |

# Feature Selection:

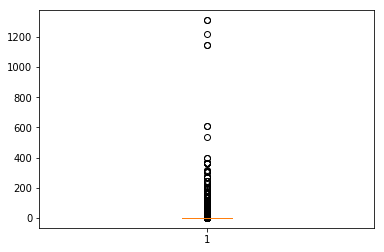
I selected my features by first selecting 28 features which are highly correlated with dependent variable. Comm24 has a correlation of 0.5 with dependent variables. So, I kept 28 variables like these. Then, I found correlation among theses 28 features. X.2 and X.18 were highly correlated and keeping both of them to explain dependent variable is not right. So, Ieliminated features which were highly correlated with each other.

After this, I was left with 13 features which are:

shares , X.5 , X , Returns , X.22 , X.11 , X.13 , commBase , X.24 , X.8 , X.2 , diff\_24.48 , comm24 .

# Exploratory Analysis and Outlier Treatment:

I ran a summary on the data for these 13 features and dependent variable. The independent variables with very large distance between 3rd quartile and maximum values were explored.

****

The variable X above has some observations that are far beyond range of others. It might be a mistake.

Hence, these observations had to be removed from the total dataset (train + test dataset) to prevent impact to model.

Similarly, other variables with large ranges had skewed distributions which implies outliers are possible. So, I removed some observations which were considered outliers.

# Task 1: Partition dataset randomly into train and test sets.

# Data was split in the following proportions using train\_test\_split function after outlier treatment.

|  |  |  |
| --- | --- | --- |
| Train | 70% | 53660 |
| Test | 30% | 22998 |

# Scaling data

The dependent variables were in variety of ranges. Hence, they were standardized using the function minmaxscaler to ensure the cost function is not skewed to the features with very large ranges.

For example: returns range from 0 to 420131 whereas X.5 range from 0 to 22. So, its important to standardize the data.

# Task 2: Use linear regression model to model the number of comments a post will receive in next H hours.

# Number of comments per hour = β0 + β1 shares + β2 X.5 - β3 X – β4 Returns – β5 X.22 + β6 X.11 + β7 X.13 + β8 commBase - β9 X.24 + β10 X.8 - β11 X.2 + β12 diff\_24.48 + β13\*comm24

# Task 3: Implement the gradient descent algorithm with batch update rule

I implemented gradient descent algorithm with batch update rule on training data with 13 features.

My initial parameter values were: learning rates (alpha) = 0.95, Number of iterations = 3000, convergence threshold = [1,1,1,1,1,1,1,1, 1,1,1,1,1]. After implementing this algorithm, my linear regression equation was:

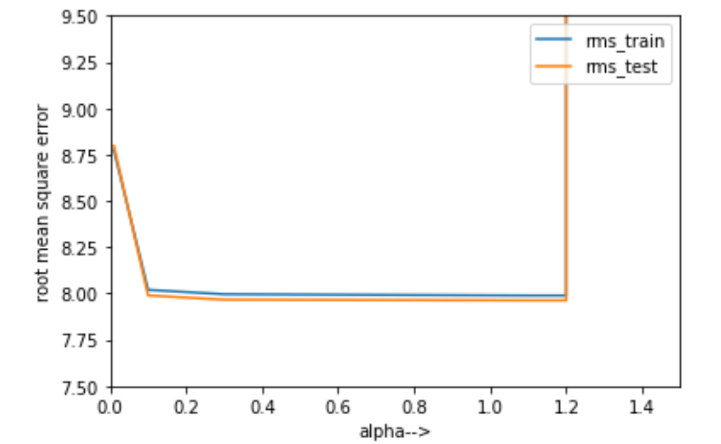
# Number of comments per hour = -4.2241 + 17.387\*shares + 2.5675\*X.5 - 1.0135\*X – 0.2098\*Returns – 0.00915\*X.22 + 2.4618\*X.11 + 3.0351\*X.13 - 9.4859\*commBase -3.7428\*X.24 + 11.857\*X.8 +0.57796\*X.2 + 9.06154\*diff\_24.48 + 48.6408\*comm24

# Experimentation 1: Experiment with various values of learning rate (∝).

I found root mean square error for train and test datasets using different learning rates.

With increasing alpha, keeping number of iterations = 3000 and convergence threshold = [1,1,1,1,1,1,1,1,1,1,1,1,1,1] ,root mean square error for both test and train datasets first decrease and then increase rapidly.

|  |  |  |
| --- | --- | --- |
| Alpha | Root Mean Square Error for train dataset | Root Mean Square Error for test dataset |
| 0.01 | 8.774703 | 8.796603 |
| 0.05 | 8.077152 | 8.061339 |
| 0.10 | 8.019494 | 7.989403 |
| 0.30 | 7.996071 | 7.966386 |
| 0.60 | 7.990577 | 7.963264 |
| **1.10** | **7.9891** | **7.9628** |
| 1.20 | 7.989058 | 7.962902 |
| 1.30 | 2.881040e+54 | 2.880097e+54 |



My best alpha will be 1.10 as it gives least root mean square error for dependent variable in both test and training data before the error starts increasing first in test set (for alpha = 1.20) and then in training set too.

# 

# Experimentation 2: Experiment with various thresholds for convergence.

For learning rate = 1.10 and number of iterations = 3000, I found root mean square error for train and test datasets using different convergence thresholds. The error first decreases and then increases for train dataset. But the error is decreasing for test dataset.

|  |  |  |
| --- | --- | --- |
| Convergence thresholds | Root Mean Square Error for train dataset | Root Mean Square Error for test dataset |
| [-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,- 5,-5,-5,-5] |  | 7.964502 |
| [0, 0,0, 0,0, 0,0, 0,0, 0,0, 0,0, 0] | 7.989141 | 7.963134 |
| [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1] | 7.989117 | 7.962887 |
| [5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5] | 7.989071 | 7.961987 |
| [10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10] | 7.989125 | 7.961061 |
| [15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15] | 7.989303 | 7.960357 |

# C:\Users\nirle\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\A0B63D7B.tmp

I have picked [5,5,5,5,5,5,5,5,5,5,5,5,5,5] as my threshold for convergence as it gives minimum root mean square error for dependent variable in both train data and test data before the error starts increasing in train set.

Keeping learning rate = 1.10 and convergence threshold as an array of 5s, I found how error varies with varying number of iterations.

# C:\Users\nirle\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\A1709CF7.tmp

After 2000 iterations, change in error for both training and test dataset is almost insignificant.

# Experimentation 3: Pick five features randomly

5 random features that I picked are:

hrs, likes, X.20, shares, tue\_pub

Keeping convergence threshold, alpha and number of iterations same for the new dataset with 5 features and original set of features, error for both train and test data sets are as follow:

|  |  |  |
| --- | --- | --- |
| Number of features | Root Mean Square Error for train dataset | Root Mean Square Error for test dataset |
| 5 (Random) | 11.391 | 11.0462 |
| 13 (Initially selected) | 7.996 | 7.966 |

Root mean square error for 5 random features is higher than 13 features I selected initially after some exploratory \_ analysis. Hence this is not a better model than previous model.

# Experimentation 4: pick five features that you think are best suited to predict the output.

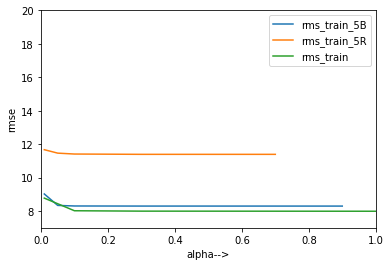
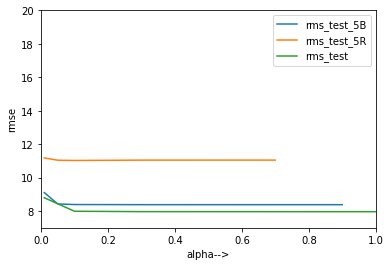
Based on exploratory analysis, 5 features ,which have the highest correlation with dependent variable, that I think are best suited to predict the number of comments in next H hours are:

comm24 , X.7 , diff\_24.48 , X.2 , X.8

Keeping convergence threshold, alpha and number of iterations same for the new dataset with 5 features and original set of features, error for both train and test data sets are as follow:

|  |  |  |
| --- | --- | --- |
| Number of features | Root Mean Square Error for train dataset | Root Mean Square Error for test dataset |
| 5 (Best) | 8.2991 | 8.3833 |
| 5 (Random) | 11.391 | 11.0462 |
| 13 (Initially selected) | 7.996 | 7.966 |

The new model with 5 features which I think are the best has root mean square error lesser than my first model but higher than the model with 5 randomly selected features for both training and test datasets. Therefore, the first model with 13 features selected initially is the best model. This has happened because the features I chose initially were highly correlated with dependent variable and affected its value a great deal. 5 random features is not better than my first model because those features had low correlation with dependent variables. And 5 best features model is also not better than 13 features model because it contains less features. And some variability in dependent variable might be explained by other features which are not included.

# Discussion:

Finally for learning rate = 1.1, convergence threshold = [5,5,5,5,5,5,5,5,5,5,5,5,5,5,5], number of iterations = 3000,

Mean squared error for training set = 7.98907

Mean squared error for test set = 7.96198

Cost function = 31.9126

I think selecting right features and learning rate matters the most for making a right model for this dataset. Randomly selecting any number of features increases error component in the model. Learning rate should be selected carefully because if it is high, we might not reach global minima.

Change in error by increasing number of iterations becomes constant after a certain number, usually 3000 iterations are enough. Convergence threshold also didn’t affect the error much.

Other steps which I could have taken to get better results with regards to modelling are taking various combinations of 10 best features to predict dependent variable or using log transformation or higher order polynomial might give better results as it narrows the variance.