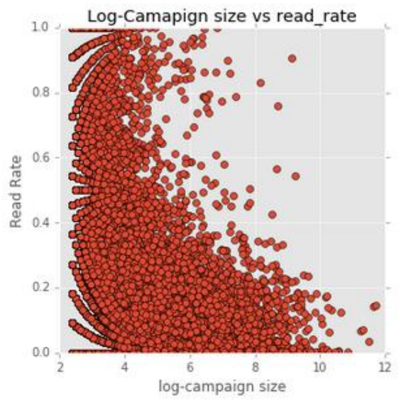
Analysis of the dataset

The dataset has 59976 rows across 19 features. There are about 25481 distinct domain hashes in the dataset.

<u>Exploratory Data Analysis</u>



The above plot is a log-campaign size vs the read rate plot.

1. Bigger campaign doesn’t imply bigger read rate
2. Smaller campaigns are more successful
3. If log-campaign size is more than 6, the likelihood that it has a read rate of below 0.6 is more.

Action Items?

1. Encourage smaller campaigns?

<u>Read Rate By Day</u>

```
Sat      0.073478
Sun      0.074181
Mon      0.096753
Fri      0.097515
Thurs    0.102786
Wed      0.102799
Tues     0.105354
```

Insights:

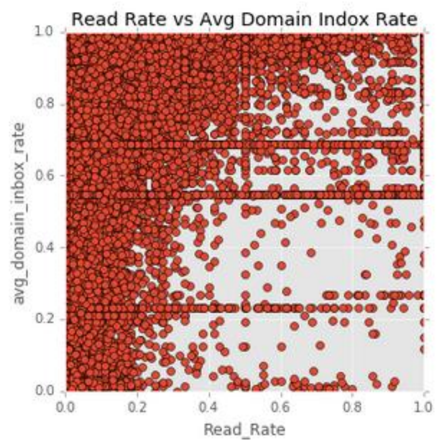
1. There is a significant increase in read rate for Tuesday,Wednesday and Thursday
2. May be due to people check their mails quite often?
3. Sending more volume on these days?

Volume of mails send per day

Sun	6840
Sat	7324
Thurs	8932
Mon	8934
Fri	9095
Tues	9283
Wed	9567

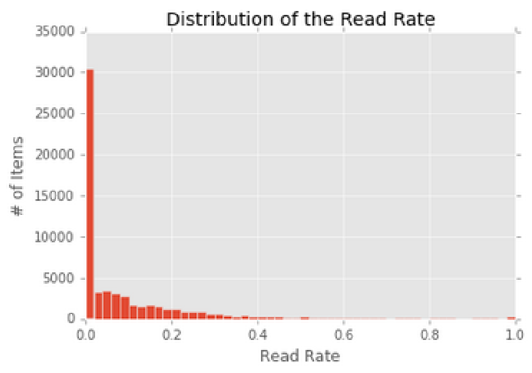
Even though we have a similar distributions of volume during weekdays, it was received better during Wed, Thu and Friday.

Read Rate vs Inbox rate



Only a fraction of mails reached to inbox is read. This seems to have no pattern.

Distribution of read Rate



Most of the mails seems to have a read rate from 0-0.1

Python Data Model

Problem statement: Given a domain hash, day and other features predict the read rate

Modelling : This was modelled as a classification problem by discretizing the read rate. The intervals from 0 to 0.4 are discretized at 0.02 intervals of 0.04 and rest till 0.5 and till 1, totally 14 segments.

Data Sampling:

Since for data, a large number of observations have unique domain hash code, samples of domain hash having less than 50 counts are removed. The rest of the data is sampled in a stratified way, so that the sampled data would have equal ratio of representation from all classes. This is included in the ipython notebook.

The data is then one-hot encoded and then SVM is trained using a grid search and three fold cross validation. The data is split as 30% Test and 70% Train. The Model performance is given as below.

We created a custom scorer to score the metrics against the un-balanced data. The accuracy is calculated at each row of confusion matrix and all their products is scored as the score of the model. This encourages to reduce false positives and false negatives.

The result of the model is as follows,

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
0	1240	465	324	153	165	101	94	62	24	36	23	24	20	16
1	78	131	48	46	20	46	6	24	15	19	14	4	10	1
2	37	49	58	29	52	27	18	9	7	9	7	9	3	1
3	23	31	15	23	7	13	15	9	11	4	6	3	5	1
4	12	13	15	9	26	11	7	2	3	9	5	6	1	3
5	3	8	4	13	5	15	5	5	7	4	6	2	1	1
6	3	4	8	6	3	3	9	3	4	7	2	7	3	1
7	0	4	0	3	2	10	5	4	7	6	3	1	1	1
8	2	0	1	2	3	4	2	3	2	3	2	0	1	1
9	3	1	1	0	0	0	2	4	2	4	0	3	3	0
10	2	2	0	1	1	3	0	0	2	3	0	2	2	1
11	1	2	1	0	1	1	3	0	1	1	0	3	2	2
12	11	4	7	1	1	4	5	4	4	9	12	14	80	43
13	0	0	0	0	1	0	0	1	1	0	0	0	15	39

The Score of this model is 4.19, Accuracy is 0.37

However this may not be the best model, it needs further feature engineering and data evaluation.

R Model

A Random forest model is used to classify the test and train splits (30% Test, 70% Train) files from python. The model is trained with 500 trees of depth 3. Cross validation could have improved parameter selection, however to start, this model works great.

Evaluation:

The confusion Matrix is as follows,

```
Accuracy : 0.620308542482155
```

```
##
##          1    2    4    5    6    7    8    9   10   11   12   13   14
##    1  2627    0    0    0    0   86   17   10    5    1    1    0    0
##    2     6    0    0    6    0    1    0    4    0    3    2    1    0
##    3     6    0    1    7    0    1    0    1    1    0    2    0    0
##    4     1    0    0   11    1    0    1    0    2    0    2    0    0
##    5     2    0    1  189    1    0    0    0    2    1    3    0    0
##    6     4    0    0   32   19    0    0    1    1    0    0    0    0
##    7   246    0    1    0    0  174   18   19    2    2    0    0    0
##    8   122    0    0    0    0   83   80   15    6    4    5    0    0
##    9    48    0    0    2    0   36   35   35    7    2    1    0    0
##   10    34    1    0    3    0   11   27   13   22    8    3    0    0
##   11    25    0    0    4    0    8    7   14    5   11    5    0    0
##   12    12    0    0    3    0    3    6    5   11   10   12    1    0
##   13    12    0    2    7    0    3    1    6    2    9    4    1    0
##   14     4    0    0    8    0    1    2    4    2    1    2    1    1
```