

GEORGIA INSTITUTE OF TECHNOLOGY

ISYE 7406: DATA MINING &
STATISTICAL LEARNING

Behavioral User Profiling in
Recommender Systems

Abhijeet Mavi
(GTID: 903518738)

Ashwin S. Pothan
(GTID: 903467532)

Sagar Tolani
(GTID: 903472774)

Instructor:
Dr. Yajun Mei

April 21, 2020



Contents

1	Project Overview	4
2	Dataset Description	5
3	Exploratory Data Analysis	7
4	Feature Engineering	9
5	Preparation of Data	11
6	Methodology	11
7	Baseline Models	12
7.1	Logistic Regression	12
7.2	Random Forest	13
7.3	Platforms for Baseline models	13
7.4	Performance Metrics	14
7.4.1	Data 1	14
7.4.2	Data 2	15
7.4.3	Data 3	16
8	New Age Recommender System: Long Short Term Memory (Recurrent Neural Network)	17
9	Above and Beyond: Behavior LSTMs	19
10	LSTM and BLSTM setup	21
10.1	Addressing Overfit	21
10.2	Optimiser, Hidden State and Loss Function	21
11	LSTM vs BLSTM: The Final Results	21
12	Results and Conclusion	24
12.1	Learning Outcomes	25
13	References	26

List of Figures

1	Pageviews dataset	5
2	Events and click dataset	6
3	Document Meta information dataset	6
4	Confidence interval of categories and topics in a document	7
5	Dist. of number of clicks by all users	7
6	Dist. of occurrence of ads (L) and click-rate for ads (R)	7

7	Dist. of browsing history by traffic_source (L) and platform (R) .	8
8	Distribution of unique documents read by user-base	8
9	Click based distribution of document features	9
10	Overview of datasets prepared	11
11	Methodology flow for models	12
12	Computing optimal λ in (0.001-0.009), (0.01, 0.09) and (0.1, 20)	14
13	Computing optimal tree-depth for Data 2	15
14	Computing optimal tree-depth for Data 3	16
15	The basic recurrent neural network cell	17
16	Sometimes old information is useful	18
17	Gates of LSTM	18
18	Behavior LSTM Cell	20
19	LSTM: Data 1	22
20	Behavior LSTM: Data 1	22
21	LSTM: Data 2	23
22	Behavior LSTM: Data 2	23
23	LSTM: Data 3	24
24	Behavior LSTM: Data 3	24

List of Tables

1	Performance metrics for Data 1	14
2	Performance metrics for Data 2	15
3	Performance metrics for Data 3	16
4	Final Results of LSTM and BLSTM for all datasets	24

Abstract

The peculiar problem of recommendation systems has encapsulated various industries with their far reaching applications in the new age of media, e-commerce and e-services like social media and ad services. The datasets are not only huge and hard to handle, but they generally have missing values and false values which make the data even more intractable. One of the organisations that provide and suggests such services on the internet is Outbrain which uses user data from different domains along with the advertisement metadata recommends advertisements to a user which are more clickable. The users in itself, have a role to play in this recommendation and hence profiling a user based on his/her salient features is what recommender systems try to achieve. We build a recommender system based on the subconscious mind of a user by taking inspiration from the state-of-the-art Long Short Term Memory recurrent neural network and profile users' subconscious mind based on his/her views of a document or advertisement. The objective of this project is to handle a big data and then apply principles of recommender systems to profile a user efficiently.

1 Project Overview

Recommender systems are becoming ubiquitous in today's world, with applications not limited to consumer goods, research articles, and financial services. An important aspect in every recommender system is its model of user preferences in order to identify the needs and interests of an individual user. In this project, we will dig deep into these systems utilizing the user's nature of interests based on their behavior as the building blocks [1].

For this project, we have selected a competition dataset from Kaggle titled 'Outbrain Click Prediction' [2]. The dataset provides the browsing history of multiple users over a 2-week period, the content type, publisher and other details about the webpage that the user is browsing. Further, details about recommendations i.e. links to similar articles on the current webpage are provided, along with the recommendation chosen by the user. The objective is to estimate which recommendation the user will click out of the given recommendations on the webpage.

In addition to utilizing the user's behavior such as clicks and ratings, we will look at modelling the user's temporal granular interactions (henceforth, called as micro behaviors). These micro behaviors build the next level of abstraction for a particular user, giving them stronger individual identity as a consumer in the system. This, in turn, will enhance the quality of recommendations.

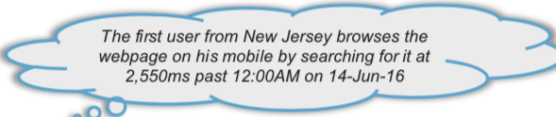
Our motivation for working on this project is two-fold. First, through the process of understanding the dataset and behaviour LSTMs we hope to gain valuable insights into the workings of Recommender Systems and Deep Learning concepts which we're sure would help us in our future professional and academic careers. Second, the availability of a well-structured data allows us to perform various data operations to create distinct datasets and analyze their performance, helping us to tackle different kinds of Recommender System applications going forward.

Additionally, we as a group are passionate about a particular application of Recommender Systems which is using user profiling models to recommend educational articles to school students to facilitate holistic learning. With the expanse of material and vivid explanations available on the internet, it is a remiss that only some students actually benefit from them. We consider this project as the first step towards that goal, and seek collaborations with equally motivated individuals/groups.

2 Dataset Description

The dataset can be primarily divided into 3 parts:

1. Pageviews: is a log of the webpages visited by users over the 2-week period.



uuid	document_id	timestamp	platform	geo_location	traffic_source
53ca43b326638b	1773952	2550	2	US>NJ>501	2
21f3a405b1699d	1262700	6393	2	US>OH>515	1
47d84d241e270d	1167084	9765	1	US>NY>501	1
fbdf1bff678b3b7	1340752	19184	2	US>FL>656	1
c745cf53f15066	1467277	21382	1	US>CA>803	1

2,034,275,448 rows (~90GB)

Figure 1: Pageviews dataset

It contains the following fields:

- *uuid*: User ID for 370k unique users
- *document_id*: unique ID for a webpage (referred as document)
- *timestamp*: milliseconds since first log on 14-Jun-16
- *platform* – device used for reading the document (desktop-1, mobile-2, tablet-3)
- *geo_location* - (country - state - media region(DMA))
- *traffic_source* – how the user arrives at a document (internal ads-1, search-2, social-3)

2. Events: is a collection of webpages where the recommendations given to a user and the user's choice are recorded.

The information is extracted using the following methodology:

- The first dataset in Figure 2 called 'Events' is a subset of pageviews with a unique *display_id* assigned to each row. The *document_id* field refers to the webpage being currently read by the user while the recommendations are provided.
- Using the *display_id* field, we can link the 'Events' data with the data on bottom-left in Figure 2 titled 'Clicks Train' which provides information about the recommended ads (*ad_id*) on the webpage, and the ad chosen by the user based on the *clicked* field.

display_id	uuid	document_id	timestamp	platform	geo_location
1	cb8c55702adb93	379743	61	3	US>SC>519
2	79a85fa78311b9	1794259	81	2	US>CA>807
3	822932ce3d8757	1179111	182	2	US>MI>505
4	85281d0a49f7ac	1777797	234	2	US>WV>564
5	8d0dae4bf5b56	252458	338	2	SG>00

23,120,126 rows (~1.5GB)

display_id	ad_id	clicked
1	42337	0
1	139684	0
1	144739	1
1	156824	0
1	279295	0

ad_id	document_id	campaign_id	advertiser_id
1	6614	1	7
2	471467	2	7
3	7692	3	7
4	471471	2	7
5	471472	2	7

Figure 2: Events and click dataset

- Using the *ad_id* field, we can link the bottom 2 datasets, which gives us information about the webpages (*document_id*) that each ad leads to. There is further information about the advertiser and campaign in terms of IDs.
3. Document Meta: provides information about the features of the webpages (*document_id*).

document_id	source_id	publisher_id	publish_time
1595802	1.0	603.0	2016-06-05 00:00:00
1524246	1.0	603.0	2016-05-26 11:00:00
1617787	1.0	603.0	2016-05-27 00:00:00
1615583	1.0	603.0	2016-06-07 00:00:00
1615460	1.0	603.0	2016-06-20 00:00:00

Figure 3: Document Meta information dataset

- The dataset in Figure 3 is called 'Document Meta Information' which provides us the *publisher_id* (publishing house (ex. CNN, Fox News)), the *source_id* (the specific segment that the content was published in (ex. CNN-History, CNN-Travel)) and the *publish_time* (sparse, available for only 1/3 of documents).
- The datasets in Figure 4 are the Outbrain generated confidence levels of the presence of a category or topic in a webpage.
- The confidence levels for presence of entities in a webpage are also provided, but this wasn't incorporated in our models owing to it's extremely sparse nature and computational difficulties.

document_id	category_id	confidence_level	document_id	topic_id	confidence_level
1782590	1608	0.715721	1782590	260	0.071256
1782590	1511	0.054457	1782590	113	0.020727
455149	1305	0.920000	1782590	281	0.015765
455149	1608	0.070000	455149	89	0.276621
925820	1302	0.920000	455149	260	0.066864

Figure 4: Confidence interval of categories and topics in a document

3 Exploratory Data Analysis

Based on the description of fields and files, we established that the given data is a well structured and extensive dataset. Our next objective was to identify meaningful data for the purpose of user profiling.

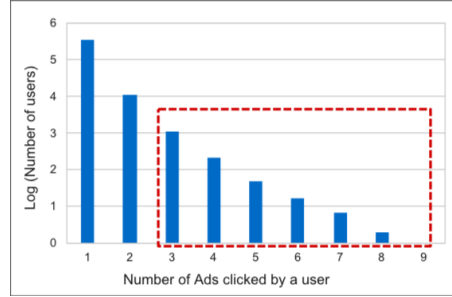


Figure 5: Dist. of number of clicks by all users

Figure 5 shows the log of number of users vs. their clicks in the dataset available to us. It was observed that over 99% users have lesser than 3 clicks which isn't meaningful in the objective of user profiling. Hence, we proceed with users with atleast 3 clicks, giving us a total of 1,352 unique users.

Now that we have selected the user-base for our analysis, the next step was to understand the interaction of these users with recommendations. We begin by understanding the quality of recommendations and user responses.

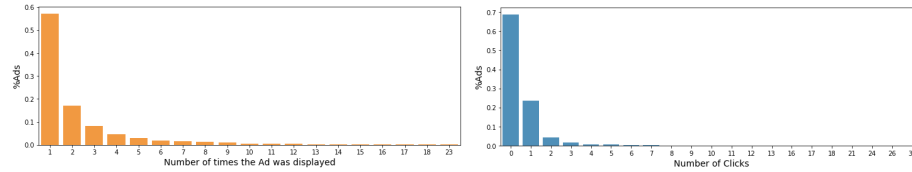


Figure 6: Dist. of occurrence of ads (L) and click-rate for ads (R)

Figure 6(L) shows the proportion of ads vs. the number of times they were

displayed. It is observed that over 80% ads appear ≤ 3 times, which can be inferred as the recommendations are user-specific.

Figure 6(R) shows the proportion of ads vs. the number of times they were clicked. It is observed that over 65% ads are never clicked on and over 20% ads are just clicked once. This brings us to the inference that the clicks are based on the user-specific taste.

Both the above inferences helped us understand the strength of Outbrain's Recommender System. Further we analyze the browsing history by traffic_source and platform.

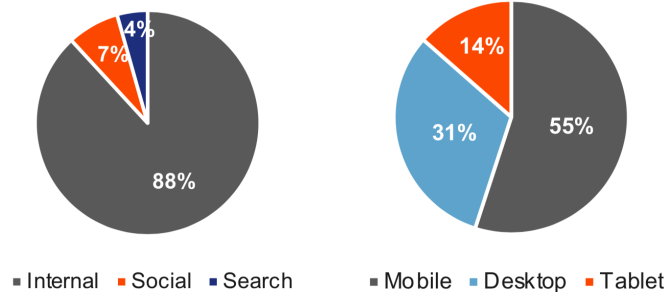


Figure 7: Dist. of browsing history by traffic_source (L) and platform (R)

Figure 7(L) shows that 88% of browsing during the 2-weeks is through internal recommendations which highlights the importance of quality recommendations to ensure that users keep finding meaningful webpages based on their tastes.

Figure 7(R) shows that a majority of browsing is through mobiles. Since all platforms have a significant contribution, we consider the feature in our model and understand the significance of device in user profiling.

Next, we proceed to analyze the reading pattern for users i.e. if a particular user visits an already read document or if the user explores new documents.

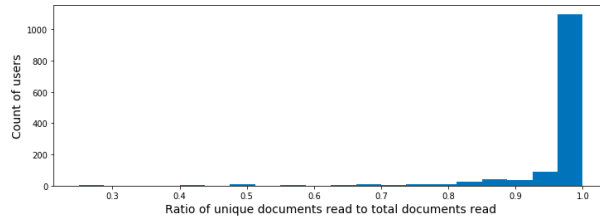


Figure 8: Distribution of unique documents read by user-base

Figure 8 shows the number of users vs. the ratio of unique documents to total documents. The peak around 1 suggests that majority of the users in the selected user-base are exploring the internet for new articles that pique their interest. It was observed that 74% of the users never read a read document again in the 2-week period.

Now that we had a basic idea about the reading pattern of the users in our user-base, we proceed to understand the impact of various document features such as advertiser and campaign of the ad, and source and publisher of the ad-document.

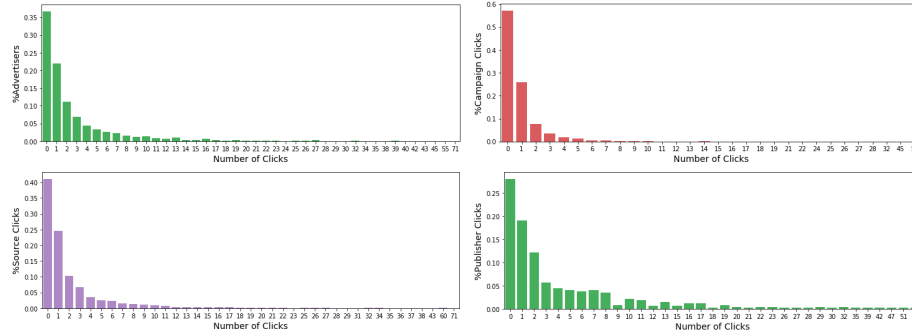


Figure 9: Click based distribution of document features

Figure 9 shows that all document features have peaks at 0-clicks followed by lower distribution of features with higher clicks. Even though the distribution doesn't provide us a good understanding of how features are relevant for ad-clicks, it provides us the information that some features have multiple clicks and we shall use these features in the model and understand their significance.

4 Feature Engineering

From the above sections, we have the following inferences which would be critical in preparation of data for our models:

- In Section 2, we observed the existence of many IDs namely *uuid*, *document_id* and *publisher_id* among others and other categorical variables such as *traffic_source* and *platform* would require one-hot encoding before they're fed into the model
- In Section 3, we observed the reading pattern of users, and distribution of clicks based on document features

Using the above observations and with the aim of providing a reasonably-sized yet meaningful dataset for our models, we were motivated to explore fea-

ture engineering. We engineered the features based on following categories:

- Time based: with the intuition that time of browsing influences the user's preference
 1. *Timebins*: signifying 4 quadrants of the day
 2. *Weekday*: whether the user browses on a weekday or weekend to understand if there's a significant difference in user interaction
- User profile based: to model the individuality of the user: Eagerness of a user "to read", to read "different" or "same" documents
 1. *User_pageviews*: total documents read by the user
 2. *User_pageviews_unique*: unique documents read by the user
 3. *User_doc_read_count*: number of times the user has read a particular document
- Document based: to model the "Popularity" of a document, it's publisher and it's source
 1. *Doc_read_count*: unique users that have read a particular document
 2. *Publisher_read_count*: unique users that have read a publisher's documents
 3. *Source_read_count*: unique users that read a source's documents
- Click based: to model the click-through-rate of an ad-document and it's features, and to understand the significance of number of recommendation choices provided in the user-profile
 1. *Click_through_rate*: total number of clicks on an ad-document
 2. *Feature_clicked*: total number of clicks on an ad-feature (campaign, advertiser, source and publisher)
 3. *Display_ad_count*: number of choices that a user was provided

Now that we have defined some engineered features which would help us make the data more intuitive and representative of the underlying problem, we provide the following hypothesis. The decision of a user to click a particular recommendation is based on both conscious and subconscious decisions.

- The conscious decisions can be modelled using the browsing history of a user, or by understanding the flow of documents being read by the user using the recommendations
- We aim to model the subconscious decisions using the above defined engineered features and document features into the behaviour node of our Behaviour-LSTM cell to understand which features do have an impact on user-profiling

5 Preparation of Data

With the well-structured click-prediction data available, we had the opportunity of creating multiple datasets with the hope that we would be better prepared to tackle relevant applications, as we take this project forward.

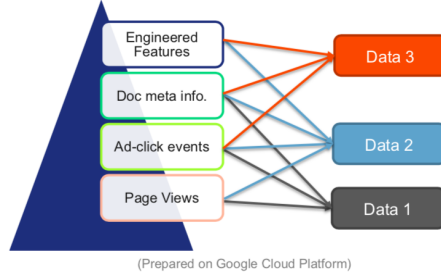


Figure 10: Overview of datasets prepared

Motivation for Data 1-2: To model click-prediction based on features of all documents read by the user-base and all the recommended documents. Here we try to model the likelihood for a document based on past reads.

- Data 1 was created as a combination of all the documents that were read by a user (Page Views), all the ads that were recommended to the user (Ad-click events) and included the document features (Doc meta info)
- Data 2 was created as an improved version by combining Data 1 with engineered features for time, user and document either read or recommended

Motivation for Data 3: To model click-prediction based on features of the recommended documents and the document being read while ad-click. Here we try to model the flow of reading pattern of a user.

- Data 3 was created as a combination the document being read currently and the recommendations provided (Ad-click events), features of document being read and being recommended (Document meta info) and engineered features for time, user, document(s) and clicks.

6 Methodology

In the following sections, we follow the methodology as described in Figure 11.

The input data was split into train set (80%), validation set (10%) and test set (10%). Training set was used to fit the model, validation set was used to tune the hyperparameters. Test set was used to get an unbiased evaluation of our model.

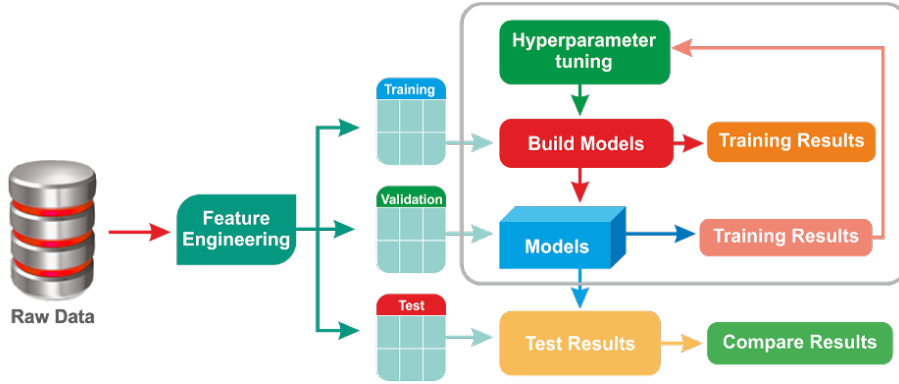


Figure 11: Methodology flow for models

7 Baseline Models

We now proceed with implementing baseline models which would help us set expectations from the more complex models. For bench-marking, we recorded three performance measures, namely misclassification rate (accuracy), run time and the computational burden required by each method.

We selected two baseline models: Logistic Regression and Random Forests

Since it's a binary classification problem, we hypothesize that the underlying distribution follows the sigmoid function, which makes applying logistic regression to time series justified for us. Also, bootstrapping in random forests can deter the order of the data and hence the algorithm might not give the best results. To signify the time sequence and maintain the order of the data, time dummies were added. These included:

1. The minute of the hour the event took place, 60 features
2. The hour of the day, 24 features
3. The particular day out of the two-week period, 14 features

In total, 98 additional features were added so that baseline algorithms work well on time series data.

7.1 Logistic Regression

Since we had binary data, clicked (1) or not clicked (0), we wanted a method that would yield the likelihood of the advertisement being clicked or not. Logistic regression being a natural binary classifier was the team's obvious choice. The coefficients are estimated by maximizing the likelihood function and the

output is squished through a sigmoid function, resulting in probability values.

Since, one hot encoding made the data sparser, regularization methods were employed to take into account important variables and to tackle overfitting problems. Namely Ridge (L2), LASSO (L1) and Elastic Net (L1 + L2) were stacked as penalty on top of logistic regression.

For L1 and L2 regularization, the optimal penalty function was found using cross validation. Whereas for the elastic net model, optimal mixing parameter (i.e. L1 ratio in scikit-learn package in python) was found using cross-validation.

The following equations represent the logistic function with the penalty function applied:

$$\begin{aligned} & \operatorname{argmax}_{\theta} \sum_{i=1}^m \log p(y_i | x_i, \theta) - \lambda R(\theta) \\ & L1 : R(\theta) = \|\theta\|_1 = \sum_{i=1}^m |\theta_i| \\ & L2 : R(\theta) = \|\theta\|_2^2 = \sum_{i=1}^m |\theta_i|^2 \end{aligned}$$

The loss function was selected as cross entropy (log loss), which outputs the probability value between 0 and 1.

7.2 Random Forest

Random forest is an improvised version of bagging which results in trees that are decorrelated. Using bootstrapped training samples, number of trees are built but at each split only a selected number of predictor variables are considered which helps in reducing variance. This would be essential for us since we are dealing with a large number of variables. To help eliminate overfitting, optimal depth of the tree was found using cross validation. Entropy was used as the loss function.

7.3 Platforms for Baseline models

For training, tuning and testing models, two online servers were used:

- Google Colab (GC) – This provided 25 GB RAM with a GPU. Since it was a completely free service, GPU type and its availability were highly variable. This was used when computational burden was low to moderate.
- Google Cloud Platform (GCP) – Free credits on a trial basis were used. A compute engine of 16 CPU's and 104 GB RAM was accessed. Used when computational burden was high.

7.4 Performance Metrics

7.4.1 Data 1

Following table represents the performance of all models on Data 1:

Model	Error rate (Validation set)	Error rate (Test set)	Run time for tuning	Server utilized	Computational burden
Logistic (Ridge)	28.88%	53.66%	6 hours	GCP	Moderate
Logistic (LASSO)	17.12%	49.07%	6.5 hours	GCP	Moderate
Logistic (Elastic Net)	17.16%	49.10%	7.5 hours	GCP	Moderate
Random Forest	32.40%	57.60%	2 hours	GC	Low

Table 1: Performance metrics for Data 1

Data 1 is the set without any engineered features. Presumably it is low in quality, which is ascertained by the high misclassification rates of all the methods. Logistic regression stacked with LASSO with $\lambda = 100$ ($\lambda^{-1} = 0.01$) performed the best out of the above four methods. Figure 12 shows how the error changes with different λ^{-1} values.

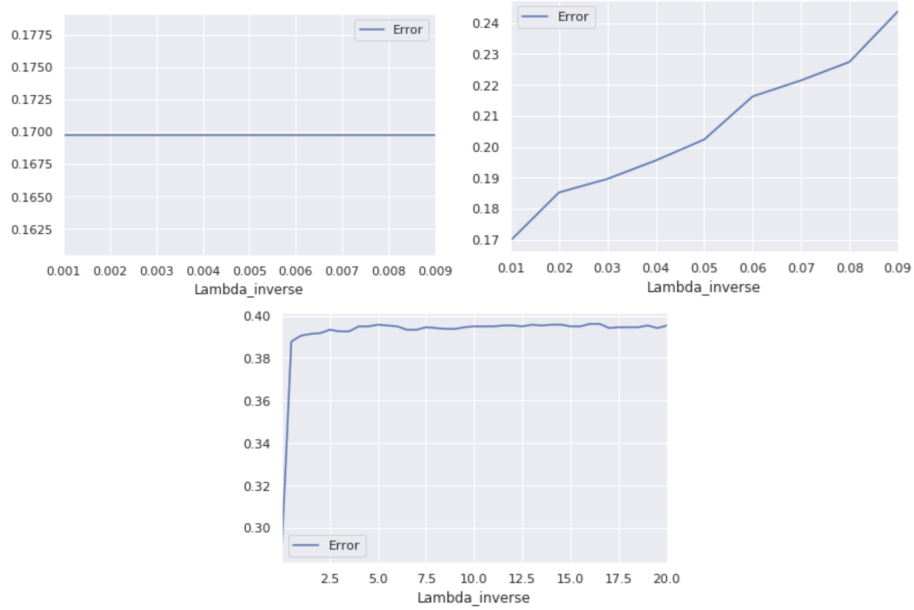


Figure 12: Computing optimal λ in (0.001-0.009), (0.01, 0.09) and (0.1, 20)

But on the downside, all the logistic regression models took long to train and deploy. Overfitting was an issue with all these models which is evident by the vast difference between the validation set error rates and test set error rates.

7.4.2 Data 2

Following table represents the performance of all models on Data 2:

Model	Error rate (Validation set)	Error rate (Test set)	Run time for tuning	Server utilized	Computational burden
Logistic (Ridge)	16.08%	16.62%	16 hours	GCP	High
Logistic (LASSO)	16.12%	16.85%	18 hours	GCP	High
Logistic (Elastic Net)	16.39%	16.65%	19.5 hours	GCP	High
Random Forest	16.20%	18.50%	3 hours	GC	Moderate

Table 2: Performance metrics for Data 2

The engineered features appreciably increased the performance and brought down the error rates between 16.62% to 18.5% for all models. Overfitting was eliminated as well.

As per the testing error rate, Logistic regression with ridge performed the best but it took longer to train and tune models (16 hours) with high computational burden. In contrast, Random forest took only 3 hours and with a moderate computing requirement. Hence in terms of all 3 performance measures, random forest was the best performing. The tree depth for random forest model was kept at 36 (found using cross validation).

Figure 13 shows how the error changes for different depth values.

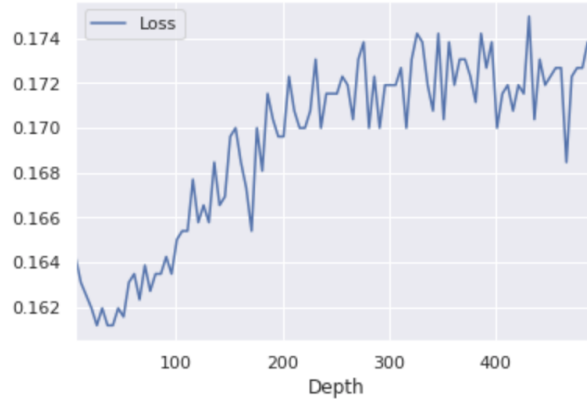


Figure 13: Computing optimal tree-depth for Data 2

7.4.3 Data 3

Following table represents the performance of all models on Data 3:

Model	Error rate (Validation set)	Error rate (Test set)	Run time for tuning	Server utilized	Computational burden
Logistic (Ridge)	16.95%	18.26%	10 hours	GCP	High
Logistic (LASSO)	16.86%	18.17%	12 hours	GCP	High
Logistic (Elastic Net)	16.78%	18.23%	13 hours	GCP	High
Random Forest	16.30%	18.26%	2 hours	GC	Low

Table 3: Performance metrics for Data 3

Random forests model was the fastest, accurate and required least computational burden. The tree depth of 86 was found using cross validation.

The plot below shows how the error changes for different depth values.

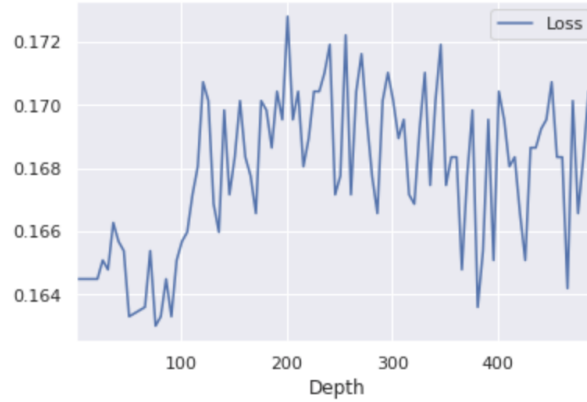


Figure 14: Computing optimal tree-depth for Data 3

Now that we have established the benchmarks for our 3 datasets, we proceed with modelling out data with LSTM and BLSTM models in the subsequent section.

8 New Age Recommender System: Long Short Term Memory (Recurrent Neural Network)

Deep learning by itself has revolutionised various industries which rely on cogent data analytics. There has been a huge uproar in terms of academic and industrial applications of deep learning.

One of the key deep learning frameworks; **recurrent neural network** has been instrumental in processing sequential methods and heuristics that build on a sequence of data or on a memory based system. The neural network is simple and at the same time easy to interpret, as what is fed into it at one time instance as an input is processed and used as a subsidiary input at the subsequent step. This is done to preserve the memory of the process as shown below.

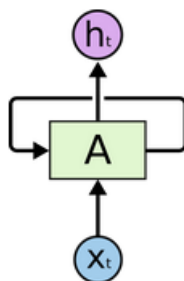


Figure 15: The basic recurrent neural network cell

The RNN is very intuitive and keeps the memory of the process whilst moving forward. The flaws are also evident as one may imagine, **remembering everything of a process ain't that important**. Keeping this memory for large systems amounts for huge computational memory and efficient coding. The unrealistic dependency on long drawn out memory might not be useful in a computational system. Sometimes, we only need to look at recent information to perform the present task. For example, consider a language model trying to predict the next word based on the previous ones. The shape of Earth is *round*. One doesn't need any further context into the statement and there is no need to keep information about Earth's size, distance from sun or it's seasons to infer this information.

RNNs can sometime need information that is not recent but was used a while back, so there are *gaps* in terms of information intake as shown below.

This provoked a need for a *filter* in the RNN cell and a better system to figure out what is needed and what isn't. **LSTMs, or Long Short Term Memory** cells of RNN are exactly that, in terms that they intuitively try to

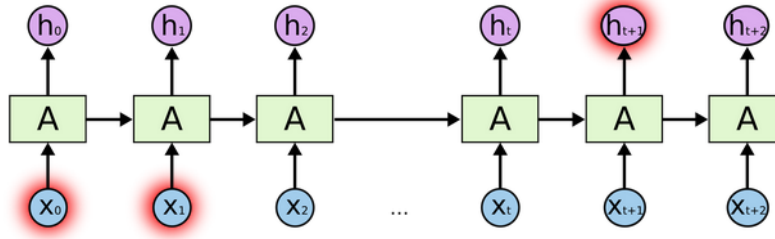


Figure 16: Sometimes old information is useful

forget the needs of the long term memory and build on the short term memory. It does so by using three efficient gates based on activation functions: **input**, **forget** and **output** gate as shown below.

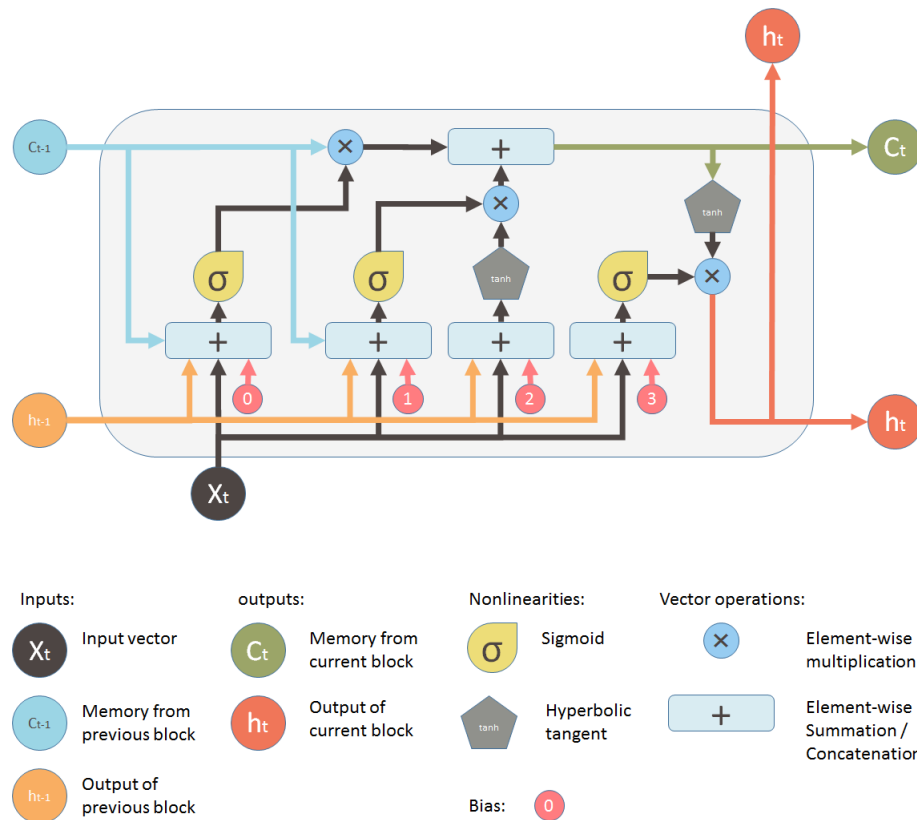


Figure 17: Gates of LSTM

The **forget gate** makes sure of what information is relevant and what is irrelevant to the procedure through the sigmoid activation function. The **input gate** provides for a weighing system to give to each feature through \tanh activation. It basically decides which values we'll update. Next, a \tanh layer creates a vector of new candidate values, \hat{C}_t , that could be added to the state. In the next step, we'll combine these two to create an update to the state. The **cell state** is like a conveyor belt that stores information on the system and keeps on adding to it as we move along the belt. Finally, the **output gate** run a sigmoid layer to decide parts of the cell state we're going to output. Then, we put the cell state through \tanh (to push the values to be between -1 and $+1$) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

The following equations govern these gates:
For forget gate,

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (1)$$

For input gate,

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (2)$$

$$\hat{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \quad (3)$$

After these two gates, the cell state gets updated as shown:

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \quad (4)$$

For output gate,

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t * \tanh(C_t) \quad (6)$$

h_t here is the hidden state that gets updated and is carried over to the next cell sequence as an input as shown in the LSTM cell diagram.

9 Above and Beyond: Behavior LSTMs

Recommender systems are a tricky problem set because sometimes the contextual information cannot be captured in terms of numbers and data. For instance, someone spending more time on a movie and re-watching a scene again and again cannot be captured efficiently through feature engineering. These actions define the **user behavior** that he or she has been working on and developing for a long time, probably far beyond the scope of LSTM-fed sequence.

These *behaviors* are not easy to track and can be lost into the sea of feature sets that are created after feature engineering, like we did in the above sections. The subjectivity of these behaviors is not captured by LSTM-RNN and needs to

be given special attention through a different heuristic. In order to account for that, we developed a novel behavior cell based on the the work done by *Yulong Gu et al* (2020) in [1]. It invokes the development of a new behavior cell built on the foundations of a LSTM-RNN cell.

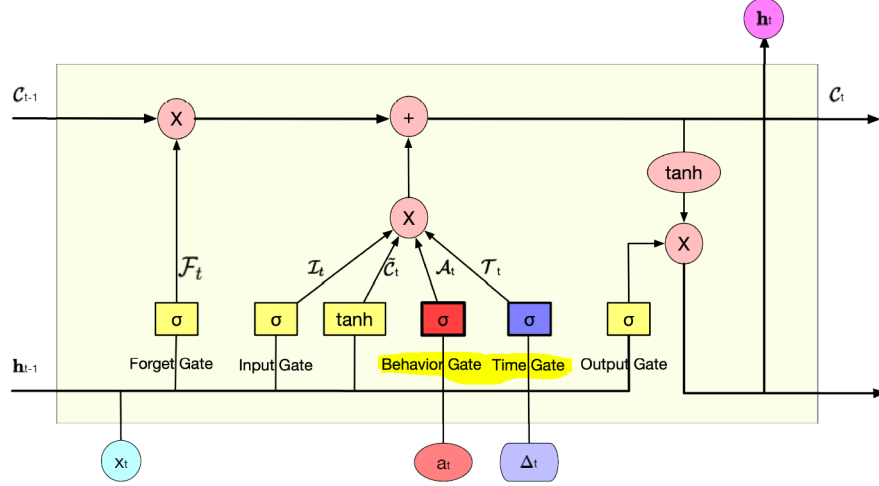


Figure 18: Behavior LSTM Cell

As one can see from the figure shown above, the LSTM includes two extra behaviour gates into the input gate. This is done using the sigmoid function and the behaviours are chosen through domain knowledge. For the two behaviours specified here, one can use the following equations for BLSTM in addition to the ones defined by LSTM,

$$a_t = \sigma(W_a[a_t, x_t] + b_a) \quad (7)$$

And, for another behavior,

$$b_t = \sigma(W_b[b_t, x_t] + b_b) \quad (8)$$

The cell state will also be update from the one defined by LSTM as follows:

$$C_t = f_t * C_{t-1} + i_t * a_t * b_t * \hat{C}_t \quad (9)$$

Specifically, what is happening here is that two feature sets are given extra priority and influence over other features and are being carried over to the next time step after due changes to them through the sigmoid activation function.

10 LSTM and BLSTM setup

10.1 Addressing Overfit

The LSTM is served with huge number of sequences in batches at a timestamp. These are prone to overfit to our existing data. In order to account for this problem, we would randomise the batches. This would work appropriately as the sequence characteristics are conserved within each batch, so an LSTM can be fitted in that region appropriately and by randomising these batches we are making sure that our fit is not being subjected to a particular long sequence but batches of data instead.

10.2 Optimiser, Hidden State and Loss Function

We used the popular **Adam** optimiser, which is an adaptive learning methodology, and the cross entropy loss to determine the misclassification rate. The **cross entropy** loss can be defined as follows:

$$Loss = -\sum_{c=1}^M y_{o,c} \log(p_{o,c}) \quad (10)$$

Here, M is number of classes (clicked, or not clicked), log - the natural log, y - binary indicator (0 or 1) if class label c, is the correct classification for observation o, p - predicted probability observation o is of class c.

We have also defined the batch size of 8 and sequence length 32 for our purposes. These numbers were taken after hyperparameter tuning to check the time taken and the error loss achieved as a metric of choosing. The values that were both powers of 2.

The computation was done on Google Colab GPU Server with 25 GB additional memory as was used for many baseline models.

11 LSTM vs BLSTM: The Final Results

Choosing the behaviors for BLSTM is something we did through intuition for our dataset. We considered how a data is constructed and hence, chose a particular behavior for the cell.

For data 1 and data 2, we considered category and topics as behavior. This is because we consider that the kind of articles being read by a user *subconsciously* builds a profile for him/her in terms that the conscious decisions can't comprehend that. Hence, a normal LSTM or any other heuristic that works on explicit data can capture that part of the data. We see from figure below for data 1,

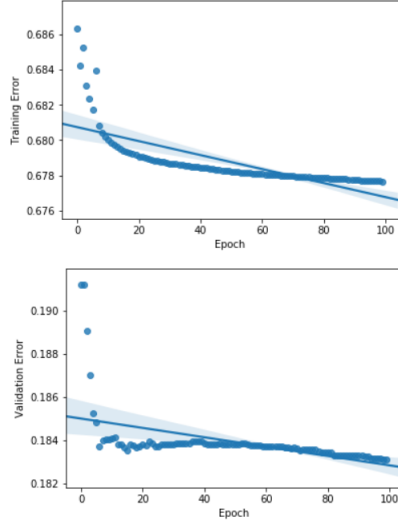


Figure 19: LSTM: Data 1

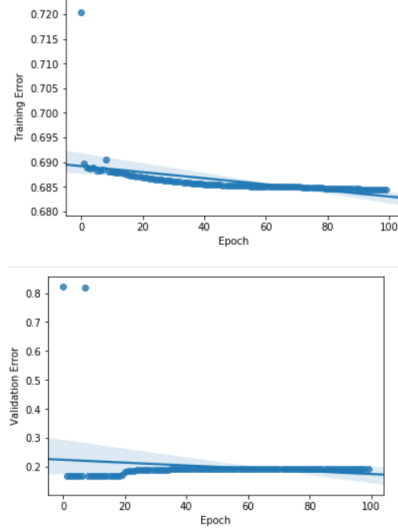


Figure 20: Behavior LSTM: Data 1

For LSTM the error does converge but it is much higher and doesn't converge as fast it does for BLSTM. This showcases the superiority with which BLSTM tracks down the results. The same exercise is done for Data 2 also and the results are shown below.

Data 2 shows a peculiar trend for training error because the error doesn't converge. Probably, because the engineered features are already compensating for better model and the model starts overfitting as epochs grow. Similarly, the validation error also, doesn't change much as the best error rate achieved through training stays as it is.

For the BLSTM, the training converges and we do get a better validation error which is constant suggesting that there isn't much change brought about through LSTM to BLSTM transition as the error are almost same. This shows that we need to account for some other kind of behavior also in order to get the better results as BLSTM has reached it's peak from this data and behavior set.

For Data 3, we chose the given behaviors of a document from its meta-data but we also considered the behaviors of a user interaction with the system. In one tuple, we considered one behavior of document's category and topics, and in another tuple we considered the number of choices that each display ID offers a particular user, since choices do effect our decision making coupled with our subconscious mind. We also took the weekday binary feature whether a user is surfing during weekday or a weekend as weekend the user might have more time to explore and might click more ads as suggested to it.

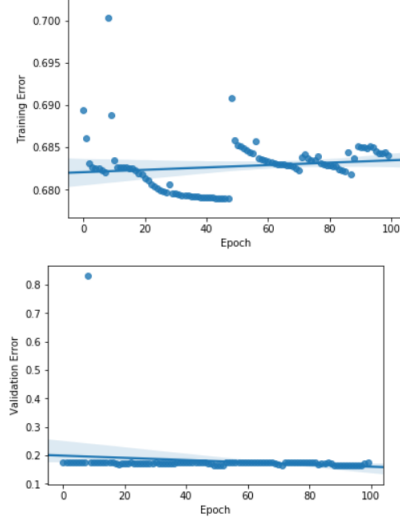


Figure 21: LSTM: Data 2

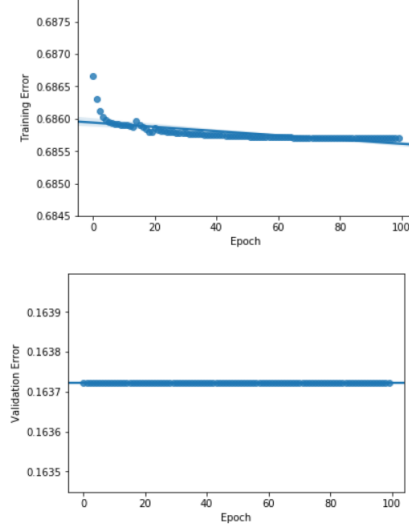


Figure 22: Behavior LSTM: Data 2

For Data 3, we see that the error rate increases as we move ahead with epochs for the BLSTM and LSTM but both converge and BLSTM definitely converges to a better error showing its superior performance. The training error rate is obviously converging for both the LSTMs as the error rate gets better with time with adaptive learning via Adam optimiser. The final results are tabulated below in Table 4.

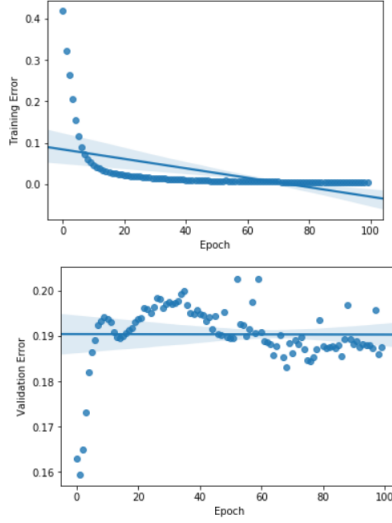


Figure 23: LSTM: Data 3

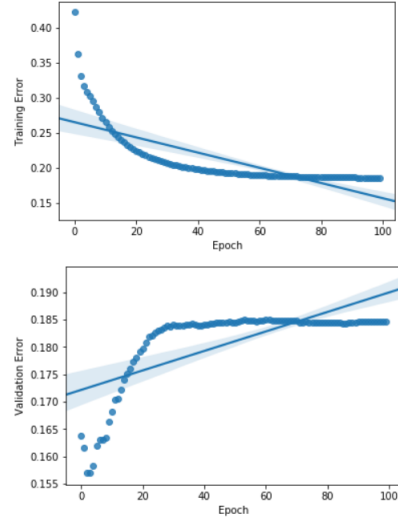


Figure 24: Behavior LSTM: Data 3

DATA	MODEL	Error rate (Validation set)	Error Rate (Test set)	Run Time	Server Utilised	Behaviors
Data 1	LSTM	18.39%	20.23%	~15mins	Google Colab	NA
	BLSTM	17.03%	17.98%	~15mins	Google Colab	"category, topic"
Data 2	LSTM	16.37%	18.29%	~15mins	Google Colab	NA
	BLSTM	16.37%	18.03%	~15mins	Google Colab	"category, topic"
Data 3	LSTM	15.87%	20.09%	~30mins	Google Colab	NA
	BLSTM	15.24%	17.56%	~30mins	Google Colab	"(Choices, weekday (0,1)), (category, topic)"

Table 4: Final Results of LSTM and BLSTM for all datasets

12 Results and Conclusion

We see that BLSTM is far superior in terms of test error rate in data 1 than all baseline model. At the same time, it is also better than the LSTM by choosing the behaviors on the document's category and topic from the given metadata.

This shows that user is building a subconscious profile based on what he is reading/clicking and this is something that is more long lasting than any of the explicit feature of a user.

The same can be said about the data 3 where we have a more refined version of data 2 with engineered features and significance put to the no. of choices given to a particular user at an instance of time. For data 2, the behavior LSTM is not able to perform because the engineered features already compensate for the forecasting and no extra information is being added the behavior feature. At the same time, a consistent convergence through BLSTM of training error shows that behaviors do provide a better stability of training to our data nonetheless. The error rate is still comparable to any of the baseline model.

At the same time, the efficiency of LSTM and BLSTM far exceeds that of any of the baseline models clearly by running on the same machine in 8-10x lesser time. This validates the original concept of LSTM where only the relevant information is taken into consideration while moving through a system. Clearly, we get faster results if only the pertinent features are considered for evaluation.

12.1 Learning Outcomes

- Through this project, we have been able to get a much deeper insight into the functioning of Recommender Systems and Deep Learning concepts
- Since we were dealing with big data (90GB), the problem was not solvable on traditional laptop CPUs, it required cloud based data processing on GPUs or TPUs on AWS and Google Colab, which was a good learning exposure for us
- The project helped us gain exposure in working with sparse datasets and encoding, and further identifying features which helped us efficiently train our base models.

13 References

[1] [Hierarchical User Profiling for E-commerce Recommender Systems](#) - Yulong Gu et al. This research was done by Data Science Lab at JD.com, and presented at WSDM '20, February 3–7, 2020, Houston, TX, USA.

[2] Dataset link: <https://www.kaggle.com/c/outbrain-click-prediction>

Various other sources:

- The Elements of Statistical Learning: Data Mining, Inference, and Prediction. By Trevor Hastie, Robert Tibshirani, Jerome Friedman
- An Introduction to Statistical Learning with Applications in R. By Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani
- [How Feature Engineering can help you do well in a Kaggle competition](#)
- [Various Kaggle Kernels](#)
- [Linked-In Learning course on PyTorch](#)
- [Understanding RNN and LSTMs - Towards Data Science](#)