상용 온라인 게임 데이터를 활용하여 잔존 가치를 고려한 이탈 예측 모델

Team name: manual_predictor

20191046 Donghyeon Lee

cheetos@gist.ac.kr

Prerequisite

For the interactive experience, execute manual_predictor.ipynb

You will need

- Python
- Numpy
- Pandas
- Matplotlib
- PyTorch
- CUDA (optional)

Performance

Rank	+/-	이름 (팀명)	제출 횟수	제출 시각	Score(Acc)
1	-	YeongWoo Nam MLDL course	3	2019-12-17 18:45:54	10564.412650
2	-	Donggun Lee MLDL 2019 Fall	3	2019-12-17 23:22:25	10175.147460
3	-	manual_predictor 개인	1	2019-12-17 21:16:36	9279.106290
4	-	Ironman 개인	5	2019-12-17 22:12:30	7957.166400
5	-	양지원 개인	4	2019-12-17 23:55:20	6163.002270
6	-	Lee Yoon Gyu MLDL 2019 Fall	1	2019-12-16 07:00:07	4048.263470
7	-	Spiderman 개인	3	2019-12-17 22:51:10	3575.651580

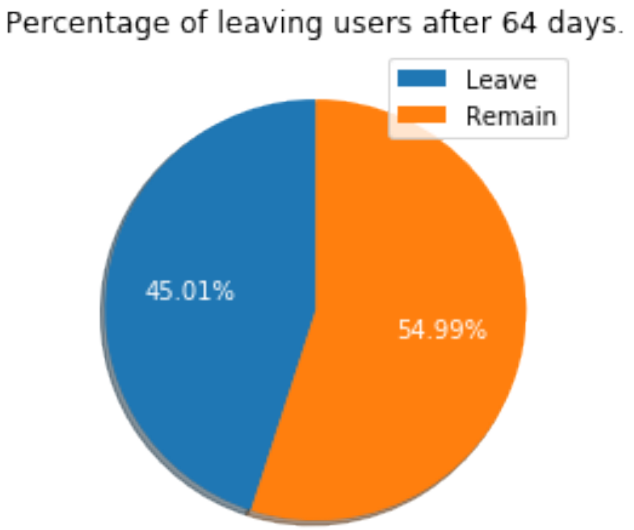
Test1: 8197.
Test2: 1081.

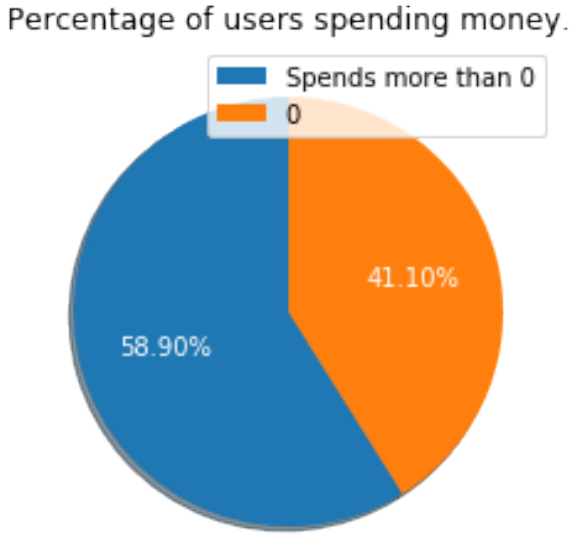
Data Distribution

- # of Leaving/Remaining users are almost 45:55.

- Also, approximately 40% of users are not spending any money.

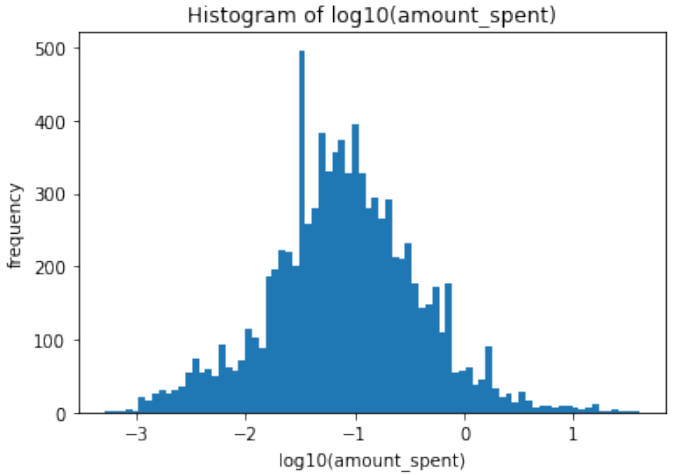
- So determining if user spending $R = 0$ and if survival time $t = 64$ would be helpful.



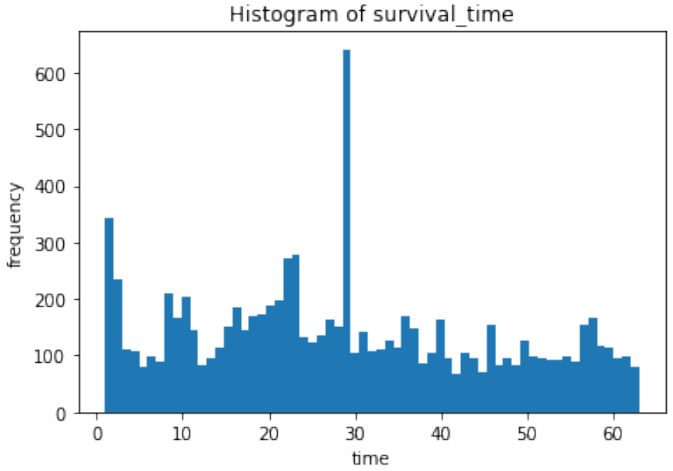


Data Distribution for $R \neq 0$ and $t \neq 64$

- Another important part is to analyze the data distribution for $R \neq 0$ and $t \neq 64$
- The distribution of values log10(amount_spent) is bell-shaped, -1 in the middle.



- The distribution of the survival_time seems random.
- But the highest peak at survival_time=29 is noticeable.



Score Function

Analyzing score function is the most important thing.

The score function is summarized by the equation below.

Based on the boundaries of the score function and the data distribution, determining if $R = 0$ and/or $t = 64$ is essential.

Also, predicting the value of R and t accurately is another objective, as you will see from the following slides.

Score function

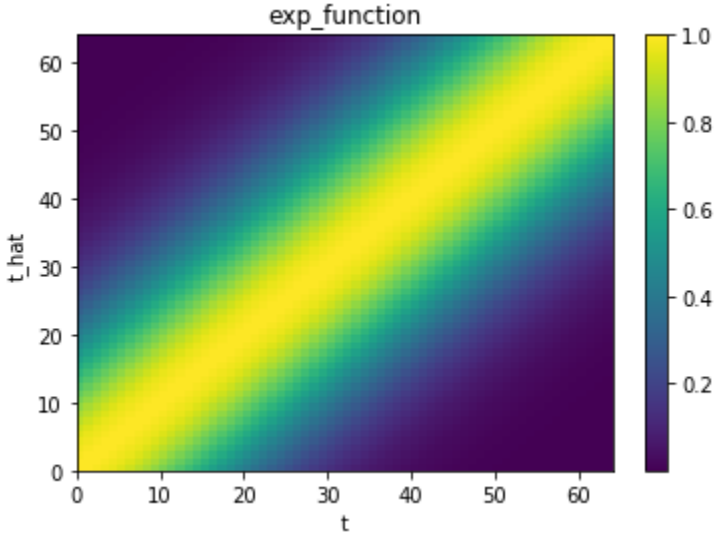
$$score^{(i)} = \begin{cases} 0 & \text{if } \hat{t}^{(i)} = 64 \\ 30\left(-0.01\hat{R}^{(i)}\right) & \text{if } \hat{t}^{(i)} \neq 64 \text{ and } (t^{(i)} = 64 \text{ or } R^{(i)} = 0) \\ 30\left(\text{clip}\left(\frac{\hat{R}^{(i)}/R^{(i)} - L}{1-L}, [0,1]\right) \times \exp\left(\frac{-\left(\hat{t}^{(i)} - t^{(i)}\right)^2}{2\sigma^2}\right) \times R^{(i)} - 0.01\hat{R}^{(i)}\right) & \text{otherwise} \end{cases}$$

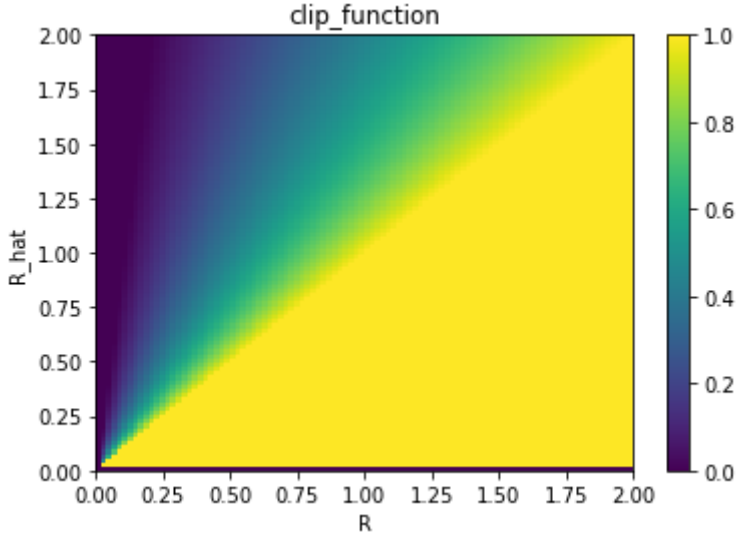
where $L = 0.1$ and $\sigma = 15$.

Score function

$$score^{(i)} = \begin{cases} 0 & \text{if } \hat{t}^{(i)} = 64 \\ 30\left(-0.01\hat{R}^{(i)}\right) & \text{if } \hat{t}^{(i)} \neq 64 \text{ and } (t^{(i)} = 64 \text{ or } R^{(i)} = 0) \\ 30\left(\text{clip}\left(\frac{\hat{R}^{(i)}/R^{(i)} - L}{1-L}, [0,1]\right) \times \exp\left(\frac{-\left(\hat{t}^{(i)} - t^{(i)}\right)^2}{2\sigma^2}\right) \times R^{(i)} - 0.01\hat{R}^{(i)}\right) & \text{otherwise} \end{cases}$$

where $L = 0.1$ and $\sigma = 15$.





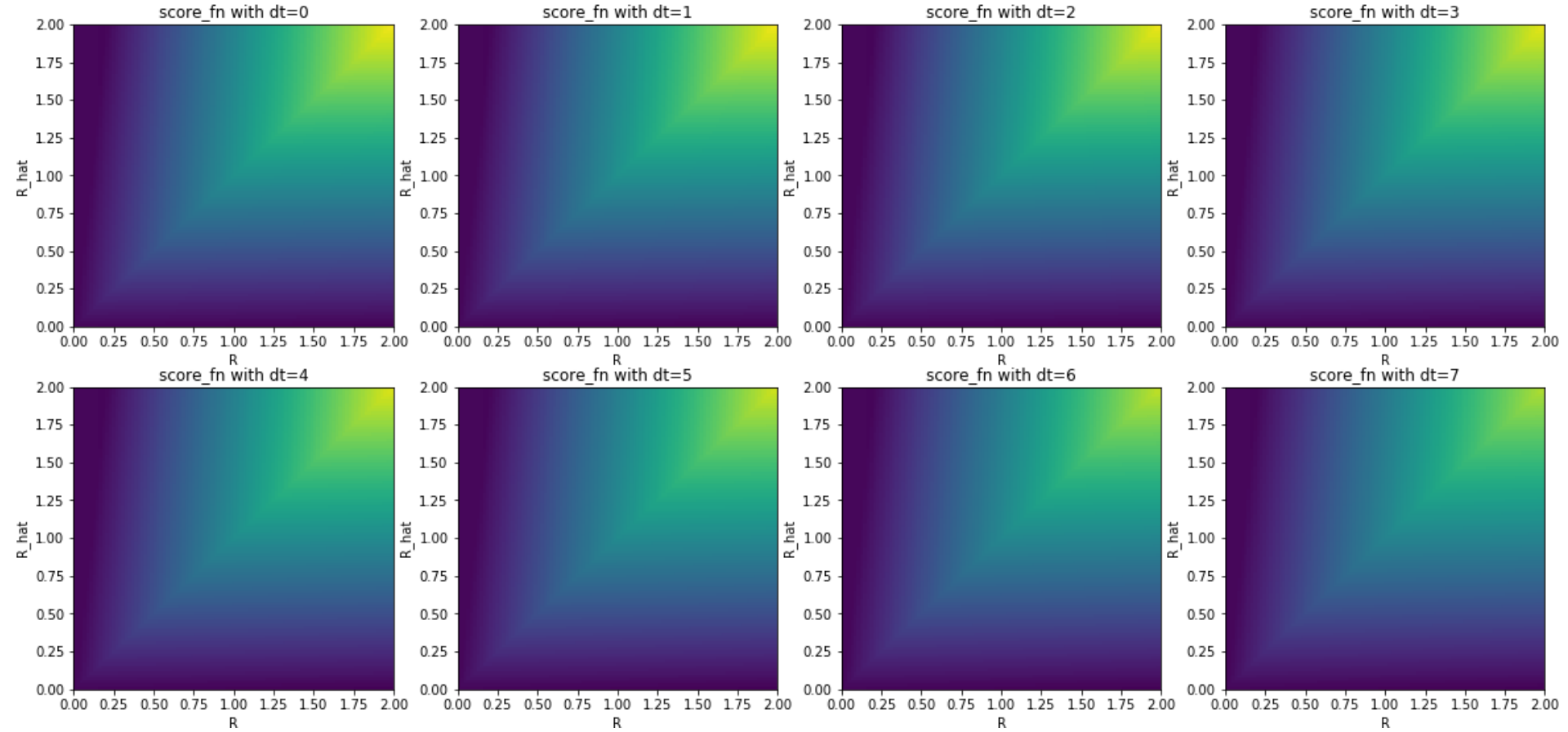
$$z = \exp\left(\frac{-\left(\hat{t}^{(i)} - t^{(i)}\right)^2}{2\sigma^2}\right)$$

$$z = \text{clip}\left(\frac{\hat{R}^{(i)}/R^{(i)} - L}{1-L}, [0,1]\right)$$

Score function

$$score^{(i)} = \begin{cases} 0 & \text{if } \hat{t}^{(i)} = 64 \\ 30\left(-0.01\hat{R}^{(i)}\right) & \text{if } \hat{t}^{(i)} \neq 64 \text{ and } (t^{(i)} = 64 \text{ or } R^{(i)} = 0) \\ 30\left(\text{clip}\left(\frac{\hat{R}^{(i)}/R^{(i)} - L}{1-L}, [0,1]\right) \times \exp\left(\frac{-\left(\hat{t}^{(i)} - t^{(i)}\right)^2}{2\sigma^2}\right) \times R^{(i)} - 0.01\hat{R}^{(i)}\right) & \text{otherwise} \end{cases}$$

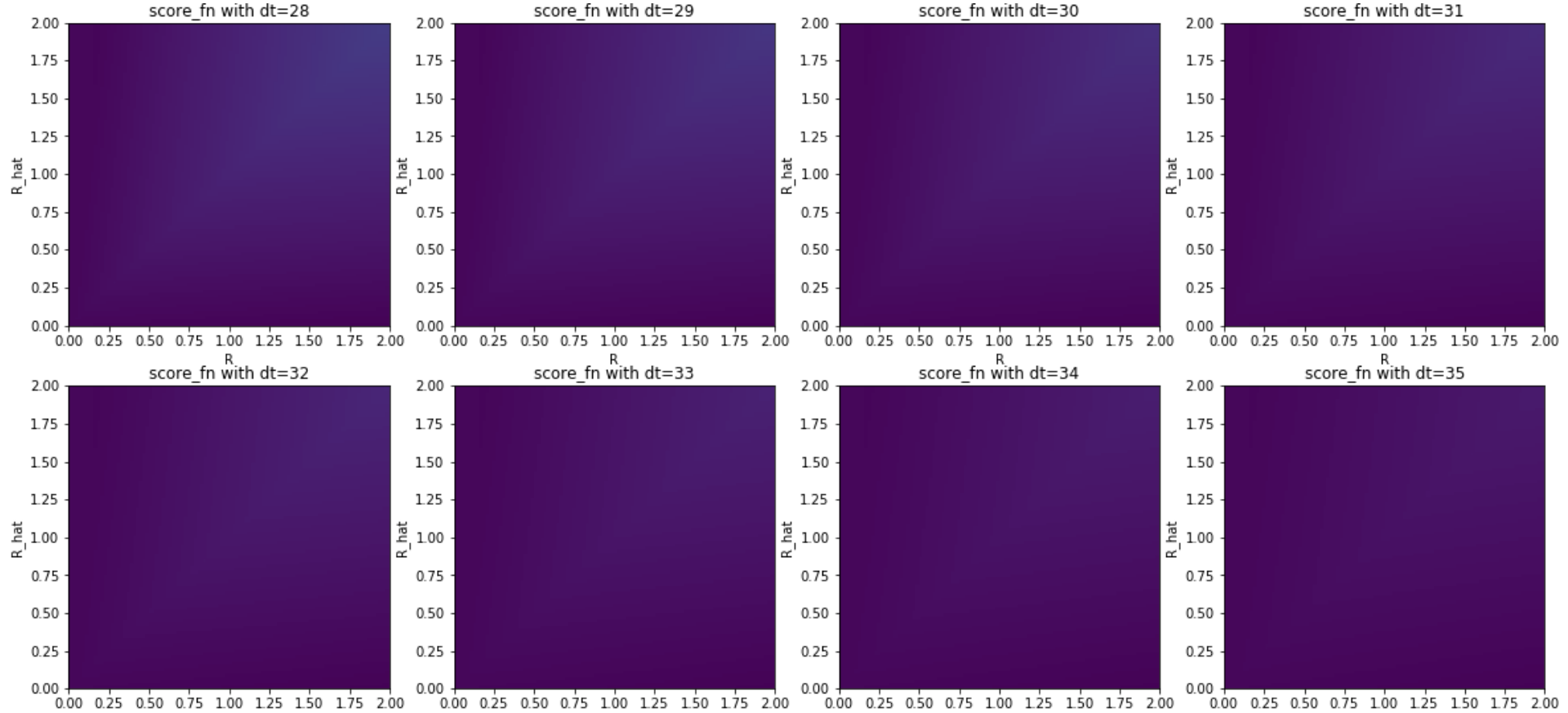
where $L = 0.1$ and $\sigma = 15$.



Score function

$$score^{(i)} = \begin{cases} 0 & \text{if } \hat{t}^{(i)} = 64 \\ 30\left(-0.01\hat{R}^{(i)}\right) & \text{if } \hat{t}^{(i)} \neq 64 \text{ and } (t^{(i)} = 64 \text{ or } R^{(i)} = 0) \\ 30\left(\text{clip}\left(\frac{\hat{R}^{(i)}/R^{(i)} - L}{1-L}, [0,1]\right) \times \exp\left(\frac{-\left(\hat{t}^{(i)} - t^{(i)}\right)^2}{2\sigma^2}\right) \times R^{(i)} - 0.01\hat{R}^{(i)}\right) & \text{otherwise} \end{cases}$$

where $L = 0.1$ and $\sigma = 15$.



Preprocessing

The “Big Picture” of the preprocessing:

1. Drop all unusable columns.
2. Merge some columns, in order to reduce the number of data.
3. Add up all the column values grouped by ‘acc_id’ and ‘day’.
4. Remove column ‘day’ by additional grouping.

Preprocessing – activity, combat, pledge

Table	Old Column	New Column
all	char_id server	(dropped)
activity	death revive exp_recovery	(dropped)
combat	class	(dropped)
combat	random_attacker_cnt random_defender_cnt same_pledge_cnt temp_cnt etc_cnt	combat_cnt
pledge	pledge_id	(dropped)
pledge	random_attacker_cnt random_defender_cnt temp_cnt etc_cnt pledge_combat_cnt same_pledge_cnt	combat_cnt_pledge

Preprocessing – trade

- Sum up the number of items (equipments, enchant_scroll, etc) and the amount of traded adena that are bought/sold from one account.
- This process generates 8 new columns.

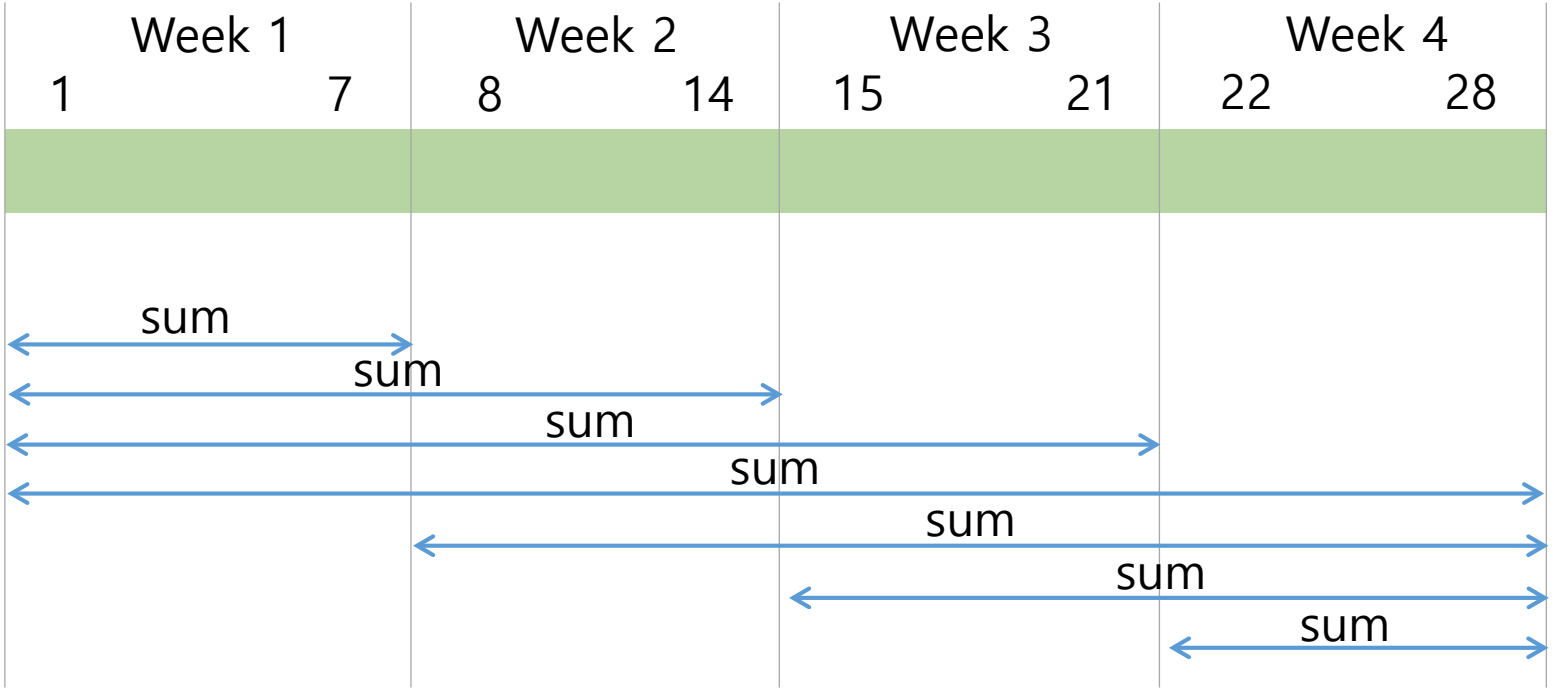
Table	Old Column	New Column
trade	time type server source_char_id target_char_id item_price	(dropped)
trade	item_type weapon armor accessory enchant_scroll adena etc	buy_equipments buy_enchant_scroll buy_adena buy_etc sell_equipments sell_enchant_scroll sell_adena sell_etc

Preprocessing – day

We set the range of days as (1) last 4, 3, 2, 1 weeks and (2) first 1, 2, 3 weeks, and extracted the data that are in the ranges.

We calculated the sum of values within the ranges respectively.

See the example on the next slide.



Example

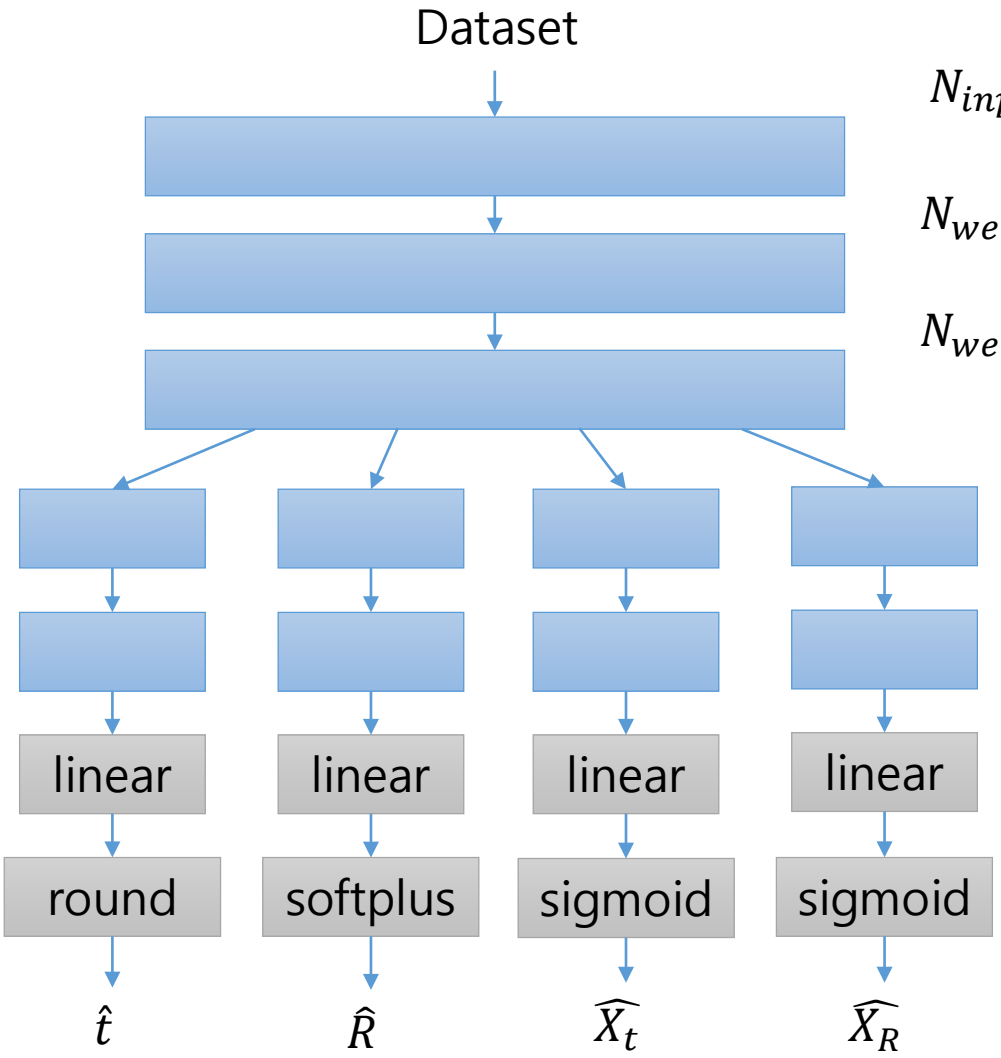
acc_id	Day	Value
1	1	1
1	2	2
1	3	3
1	8	4
1	28	5
2	…	…
…		

acc_id	1	1~2	1~3	1~4	2~4	3~4	4
1	6	10	10	15	14	5	5
2	…						
…							

Preprocessing – label

- Add 2 columns with values {0,1} to represent
  1. if the user survives after 64 days ($X_t = 1$), and
  2. if the user does not spend any money ($X_R = 1$).
- These values are trained and predicted to our model.

Model



N_{input}

N_{weight}

N_{weight}

N_{weight} → split into 4 pieces

$N_{weight}/4$ each

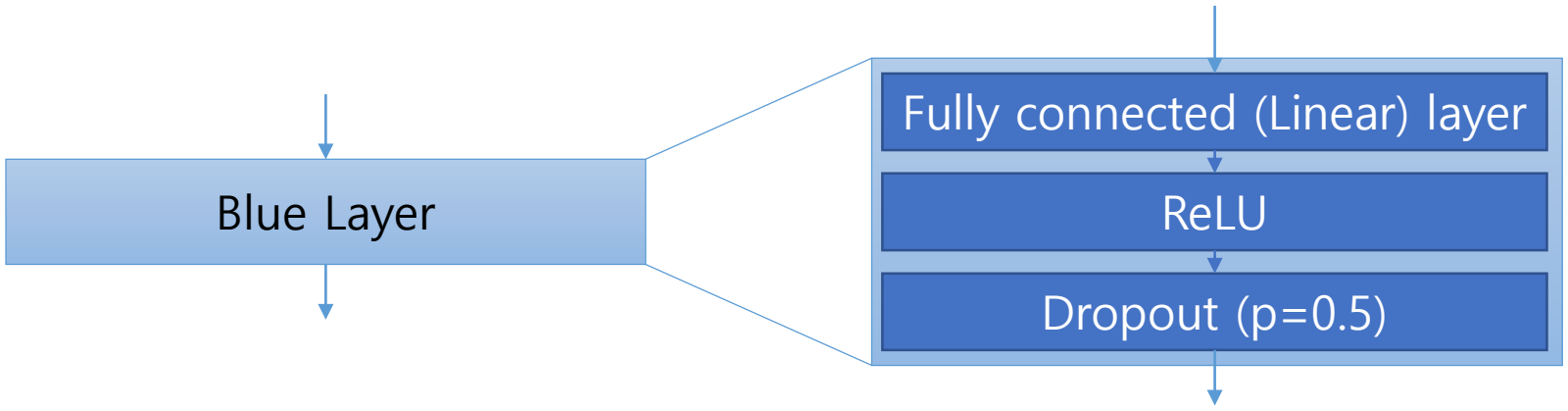
$N_{weight}/4$ each

1 each

1 each

Model Details

- linear : fully connected (linear) layer
- round : integer rounding function
- sigmoid : sigmoid function
- Softplus : softplus function
- blue layer: Fully connected + ReLU + Dropout(p=0.5)



Objective functions and hyperparams

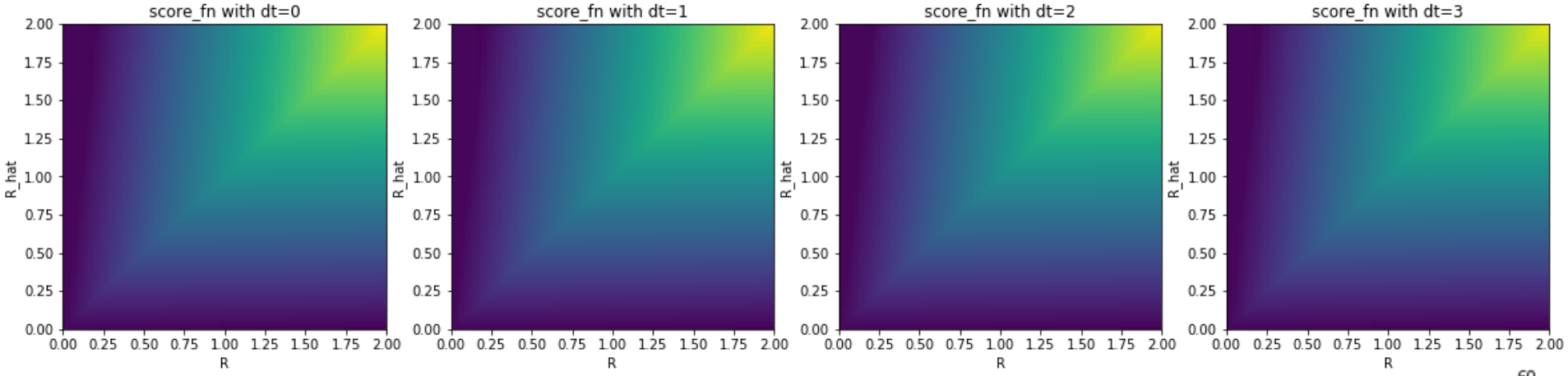
- F1 = Binary_Cross_Entropy($\widehat{X_t}$, X_t)
- F2 = Binary_Cross_Entropy($\widehat{X_R}$, X_R)
- F3 = Approximated_score_function(t, $\hat{t}$, R, $\hat{R}$)
- Weight update: Adam optimizer

hyperparams	
N_{input}	196
N_{weight}	512
Learning rate	0.05
Batch size	400
# Iterations	100

Objective functions in detail

- Score function were approximated to used in the optimization.

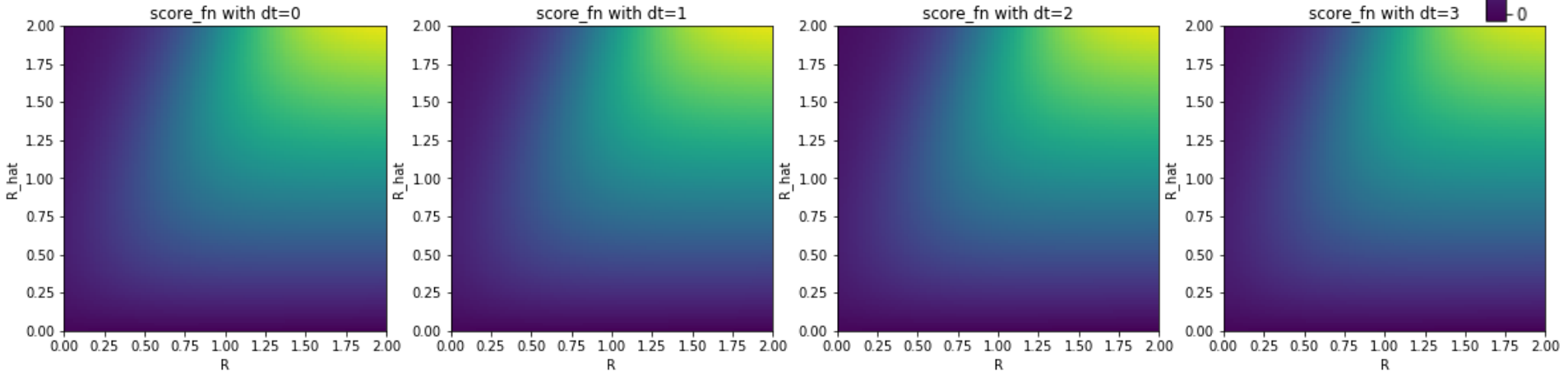
$$score^{(i)} = 30\left(\text{sigmoid}\left(4(\hat{R} - R/2)\right) \times \exp\left(\frac{-\left(\hat{t}^{(i)} - t^{(i)}\right)^2}{2\sigma^2}\right) \times R^{(i)} - 0.01\hat{R}^{(i)}\right)$$



Original score function

VS

Approximated score function



Future Enhancements

- Enhancing approximator.
- Apply log to R, to make the distribution simpler.

$$score^{(i)} = 30\left[\text{sigmoid}\left(4(\hat{R}/R - 1/2)\right) \times \exp\left(\frac{-\left(\hat{t}^{(i)} - t^{(i)}\right)^2}{2\sigma^2}\right) \times R^{(i)} - 0.01\hat{R}^{(i)}\right]$$