# ADL 2022 Team 20 Final Project

成員: 邱詠智 r10922064許智凱 r11944008, 余書齊 r11922038

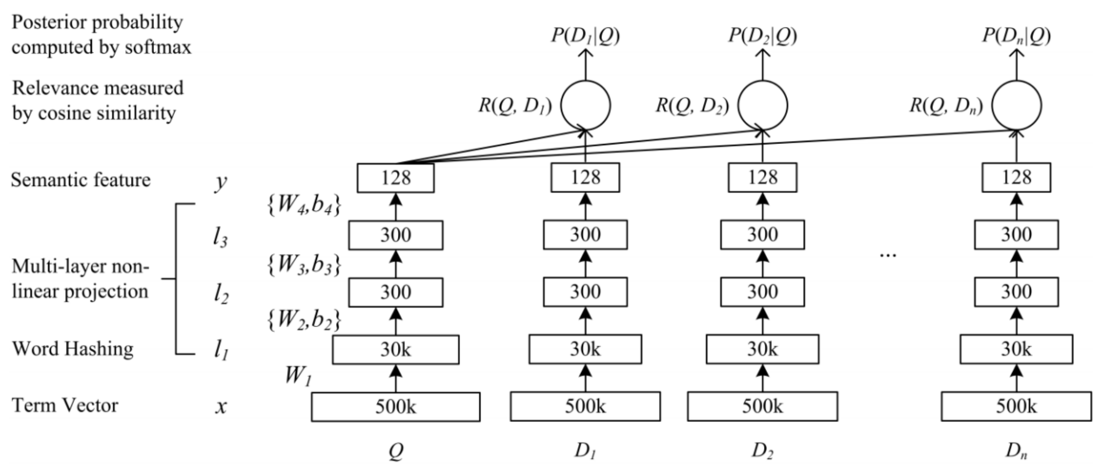
## Work Distribution

- 邱詠智: Data Analysis, 分析各 feature 的 importance.
- 許智凱: 負責 DSSM 在 seen domain 與 unseen domain 的預測以及實驗整理
- 余書齊: 負責 als 在 seen domain 與 BM25 在 unseen domain 的預測以及實驗整理

## Task Definition

- Target :
    - learn the correlation between different courses
    - predict the courses the user would buy in the future

## Related Work



### Intro

DSSM(Deep Structured Semantic Model) 是由微軟於 CIKM 2013 提出的一篇 work，該模型主要用來解決 NLP 領域語義相似度的 task，利用深度神經網路將文本表示為低維度的向量，用來提升搜索場景中 document 和 query matching 的問題。

DSSM 模型的原理主要是：通過用戶搜索行為中 query 和 doc 的數據，通過深度學習網路將 query 和 doc 映射到到共同維度的語義空間中，最大化 query 和 doc 語義向量之間的餘弦相似度，從而訓練得到隱含語義模型，即 query 側特徵的 embedding 和 doc 側特徵的 embedding，進而可以獲取語句的低維語義向量表達 sentence embedding，可以預測兩句話的語義相似度。

而在推薦系統中，最為關鍵的是如何做好 user 與 item 的 matching 問題，因此對於推薦系統中 DSSM 模型的則是為 user 和 item 分別構建獨立的子網路塔式結構，利用 user 和 item 的曝光或點擊日期進行訓練，最終得到 user 側的 embedding 和 item 側的 embedding。

- https://github.com/datawhalechina/torch-rechub

# Approach

## Data Analysis

In this section, we tried to analyze the properties of user, course, and topic datasets. It help us to realize whether a feature is important or not.

### User

The table below shows the empty ratio of each user feature.
Although there're a lot of empty users, most of them don't appear in datasets.
In each datasets ( train, val, test), every user has at least one feature, and 99.9% of them have "interests" feature.

| | **user number** | **gender** | **occupation titles** | **interests** | **recreation names** | **all empty** |
|---|---|---|---|---|---|---|
| users | 130,566 | 34.61% | 77.75% | 36.62% | 75.54% | 31.97% |
| train | 59,737 | 4.10% | 64.22% | 0.04% | 59.97% | 0.00% |
| val seen | 7,748 | 3.83% | 50.84% | 0.03% | 45.13% | 0.00% |
| test seen | 7,205 | 3.91% | 49.24% | 0.04% | 44.21% | 0.00% |
| val unseen | 11,622 | 4.26% | 69.43% | 0.06% | 66.18% | 0.00% |
| test unseen | 11,097 | 3.77% | 65.38% | 0.07% | 61.55% | 0.00% |

### Course

In this part, we analyze the relation between some course features (e.g. price, publish time) and the probability to be purchased.

The detailed definition is shown below, let $c_k$ = k-th course:

$$\begin{aligned} c_k &= (\mathrm{id}_k, \mathrm{price}_k, \mathrm{publish\ time}_k, \mathrm{other}_k) \\ \mathrm{cnt}_k &= \#\ \mathrm{of\ id}_k\ \mathrm{appears\ in\ purchase\ log} \end{aligned}$$

$$\begin{aligned} P(c_k) &= \frac{\mathrm{cnt}_k}{\sum_i \mathrm{cnt}_i} \\ P_p(x) &= \frac{\sum\{\mathrm{cnt}_k | \mathrm{price}_k = x\}}{\sum_i \mathrm{cnt}_i} \\ P_t(y) &= \frac{\sum\{\mathrm{cnt}_k | \mathrm{publish\ time}_k = y\}}{\sum_i \mathrm{cnt}_i} \end{aligned}$$

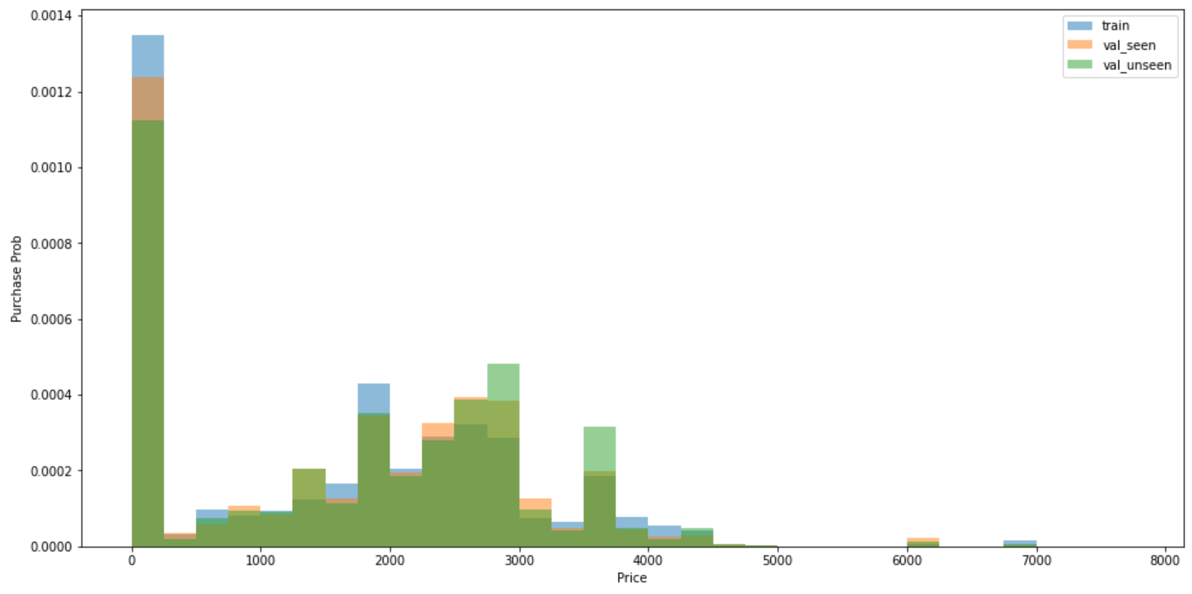
where $P_p(x), P_t(y)$ are the probability function for a course be bought with price $x$ and publish time $y$ respectively.

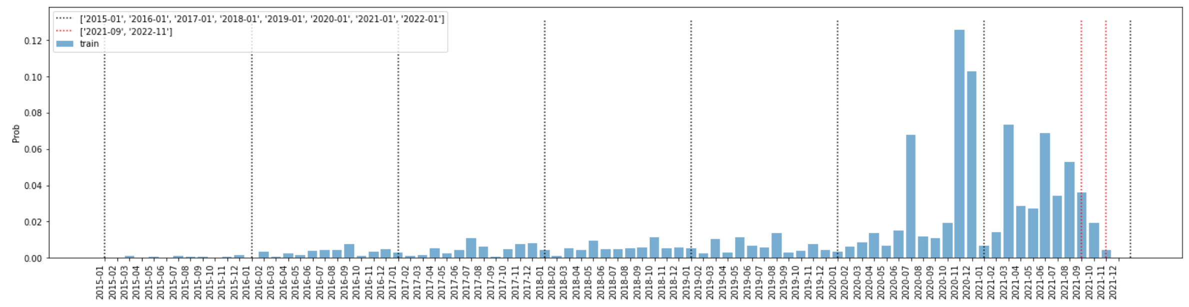
For $P_p(x)$, about 30% purchases are free course, and the distribution over train and val datasets are very similar. It's possible to approximate $P_p$ by sum of 2 normal distributions.

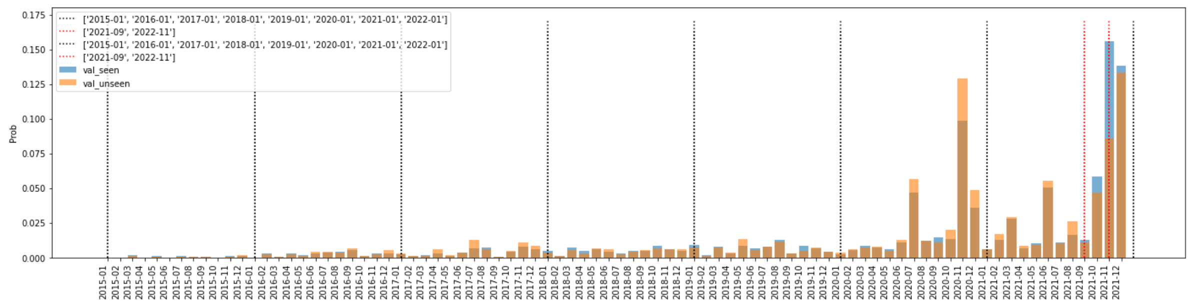
$$P_p(x) \approx [\mathcal{N}(\mu_1, \sigma_1^2) + 1.5 \times \mathcal{N}(\mu_2, \sigma_2^2)]/2$$

where $N(\mu, \sigma)$ is normal distribution with mean $\mu$ and standard deviation $\sigma$, and $(\mu_1, \sigma_1) \approx (0, 50), (\mu_2, \sigma_2) \approx (2360, 1000)$

For $P_t(y)$, the distribution over train and val dataset are very different in 2021-01 ~ 2021-12, but similar before 2021. We believe it's because of the time of data sampling.
We can "shift" or "extend" the training distrubtion from 2020-01 ~ 2020-12 to approximate the val or testing distribution.





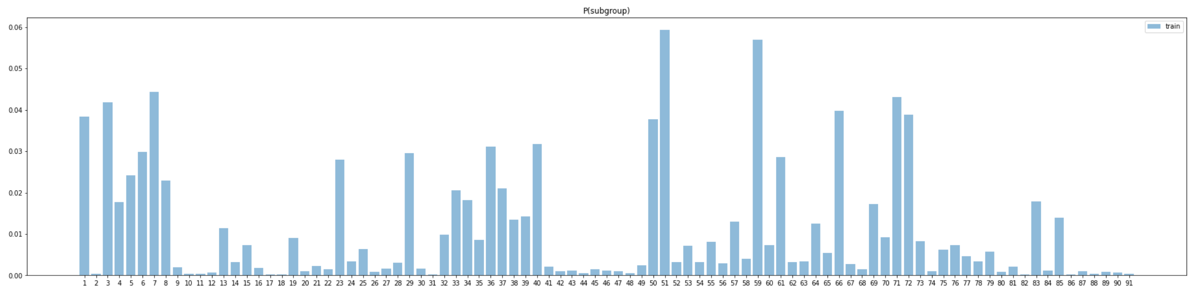


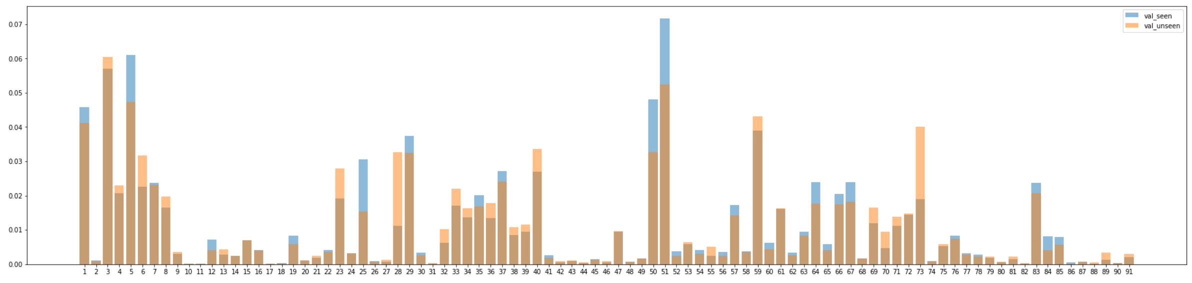
## Topic

In this part, we analyze the probability of each topic.

$$\begin{aligned} \text{topic}_k &= (\text{id}_k, \text{name}_k) \\ \text{cnt}_k &= \#\text{ of id}_k \text{ appears in dataset} \\ P(\text{topic}_k) &= \frac{\text{cnt}_k}{\sum_i \text{cnt}_i} \end{aligned}$$

The figure below shows the $P(\text{topic})$. This helps us to add some popular topics in post-processing stage.





## Baseline

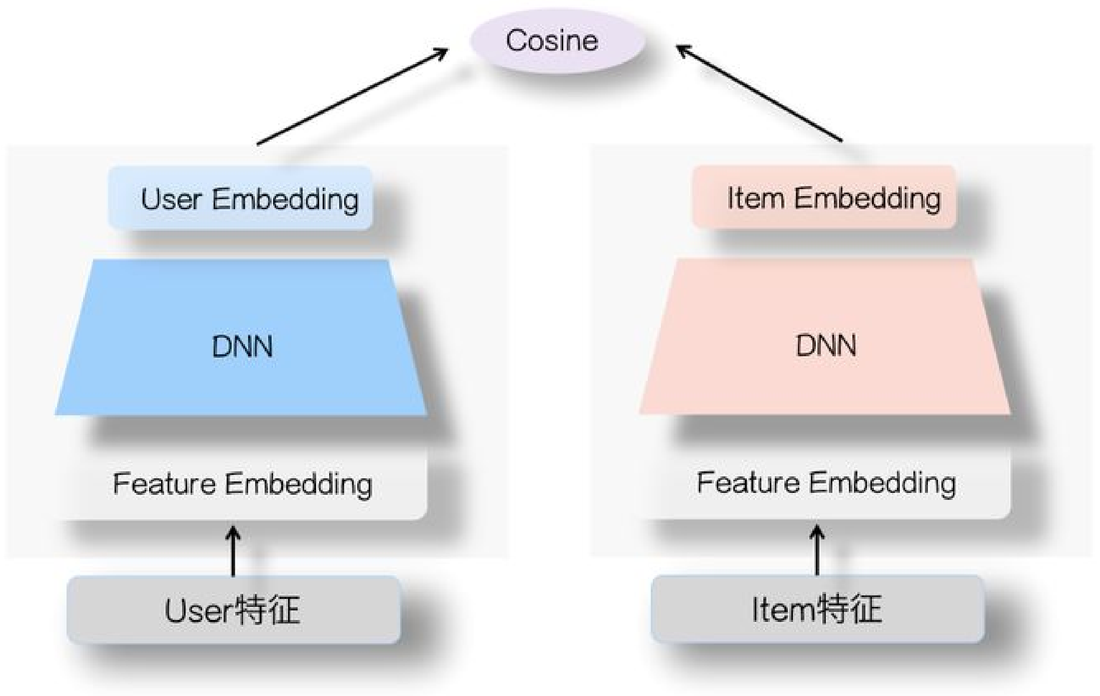
For course prediction, we use the weighted sampling without replacement as baseline model. The weight of each course is defined as:

$$\begin{aligned} w_k &= P_p(\text{price}_k) \times P_t(\text{publish time}_k) \\ p_k &= \frac{w_k}{\sum_i w_i} \end{aligned}$$

For topic prediction, we use the same strategy as baseline, where the weight of each topic is $P(\text{topic}_k)$

The kaggle score of them would be introduced in the performance section.

## DSSM Model in recommendation system



其結構非常簡單，如上圖所示，左側是 user 塔，右側是 Item 塔。

在 user 側結構中，其輸入為 user 相關特徵 ( 用戶基本信息、統計屬性以及歷史行為序列等 )；在 item 側結構中，其輸入為 item 相關特徵（ item 基本信息、屬性信息等）。

對於這兩個塔本身，則是經典的 DNN 模型，在訓練過程中，其輸入由 OneHot 到 Feature embedding，再經過幾層 DNN，兩個塔分別輸出 user embedding 和 item embedding，最後這兩個 embedding 做內積或者 Cosine 相似度計算，使得 user 和 item embedding 映射到共同維度的語義空間中。

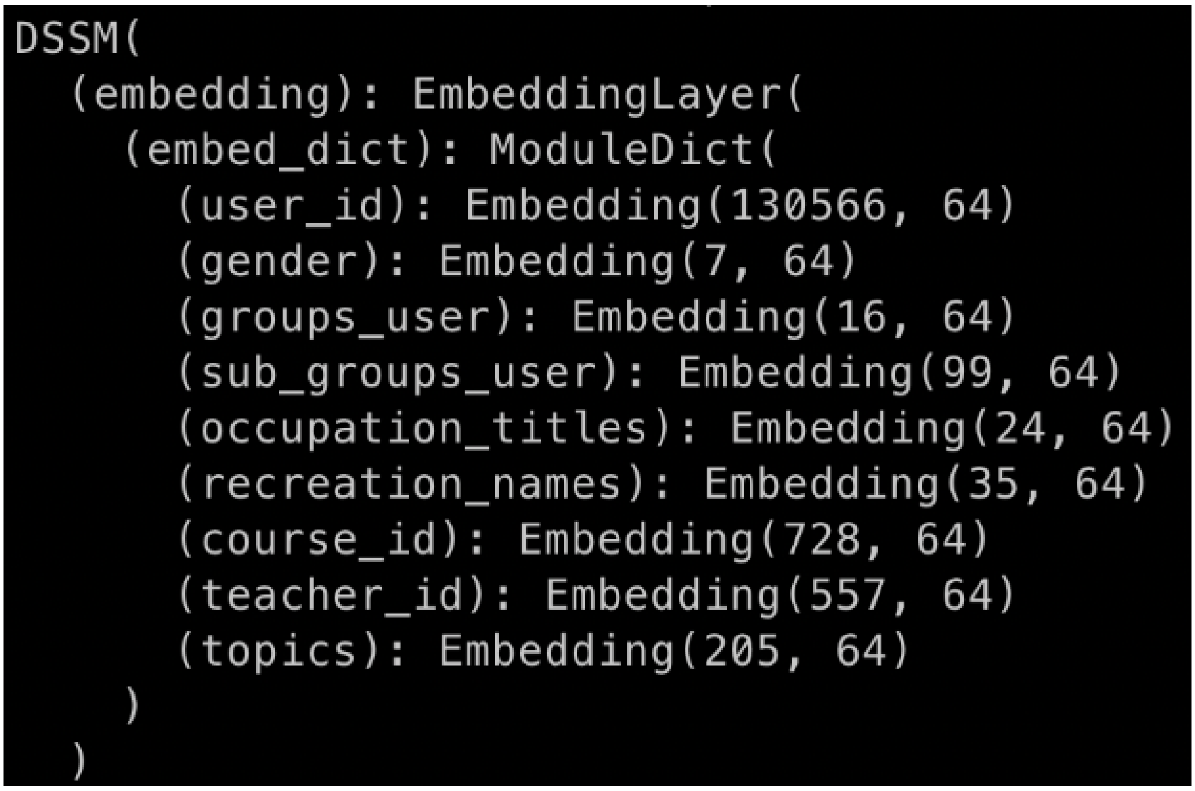
### User features

- SparseFeature
  - user_id
  - gender
- SequenceFeature: Mean Pooling
  - sequence 內的每個 element 經過 embedding layer 後，取平均
  - hist_course_id: purchase history
  - groups_user: extracted from interests
  - sub_groups_user: extracted from interests
  - occupation_titles
  - recreation_names
- Feature embedding size: 64

## Course features

- SparseFeature
  - course_id
  - teacher_id
- SequenceFeature: Mean Pooling
  - sequence 內的每個 element 經過 embedding layer 後，取平均
  - groups
  - sub_groups
  - topics
- DenseFeature: Normalize to [0, 1]
  - 希望相近的值，有相近的 embedding
  - course_price
  - course_published_at_local
  - total_sec
- Feature embedding size: 64

Note that user features `hist_course_id`, `groups_user` and `sub_groups_user` share the weights of embedding layers with course features `course_id`, `groups` and `sub_groups` respectively.



```
DSSM(
  (embedding): EmbeddingLayer(
    (embed_dict): ModuleDict(
      (user_id): Embedding(130566, 64)
      (gender): Embedding(7, 64)
      (groups_user): Embedding(16, 64)
      (sub_groups_user): Embedding(99, 64)
      (occupation_titles): Embedding(24, 64)
      (recreation_names): Embedding(35, 64)
      (course_id): Embedding(728, 64)
      (teacher_id): Embedding(557, 64)
      (topics): Embedding(205, 64)
    )
  )
```

因為 feature 是 OneHot encoding，所以 embedding 的 input  size 會是該 feature 的 maximum + 1

## Negative samples

- Global Random Sampling + Popular Suppression
  - popularity sampling method used in word2vec

對於熱門的 course 很多 user 可能會購買，需要進行一定程度的 undersampling，使得模型更加關注一些非熱門的 course。此外在進行負樣本採樣時，應該對一些熱門 course 進行適當的 oversampling，這樣可以盡可能的讓模型對於負樣本有更加 fine-grained 的區分。例如在 word2vec 中，負採樣方法是根據 word 的頻率，對 negative words 進行隨機抽樣，降低 negative words  量級。

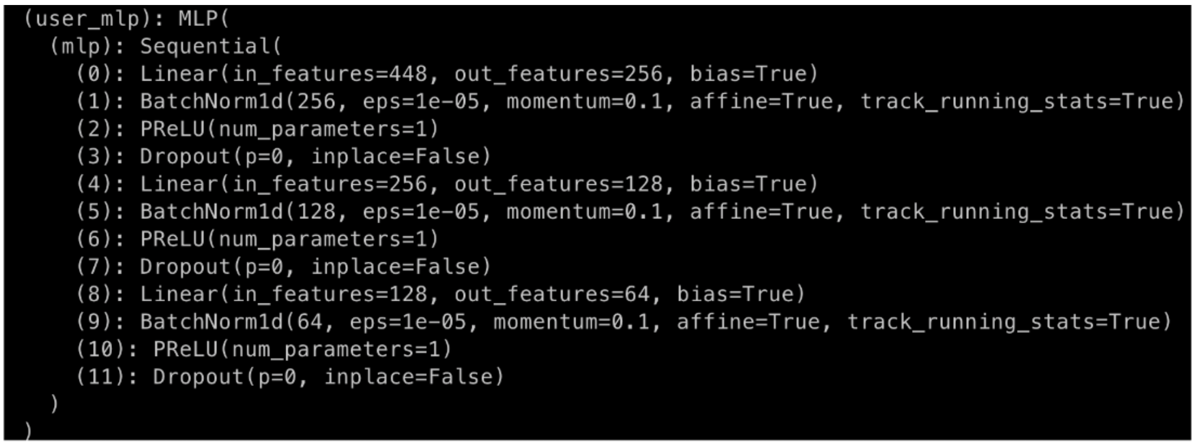
$$P(w_i) = \frac{f(w_i)^{0.75}}{\sum_{j=0}^{n} f(w_j)^{0.75}}$$

之所以熱門 course 做負樣本時，要適當 oversampling，增加負樣本難度，是因為對於全部的 course，模型可以輕易的區分一些和 user 興趣差異性很大的 course，難點在於很難區分一些和 user 興趣相似的 course。因此在訓練模型時，需要適當的增加一些難以區分的負樣本來提升模型面對相似 course 的區分能力。

## Default setting

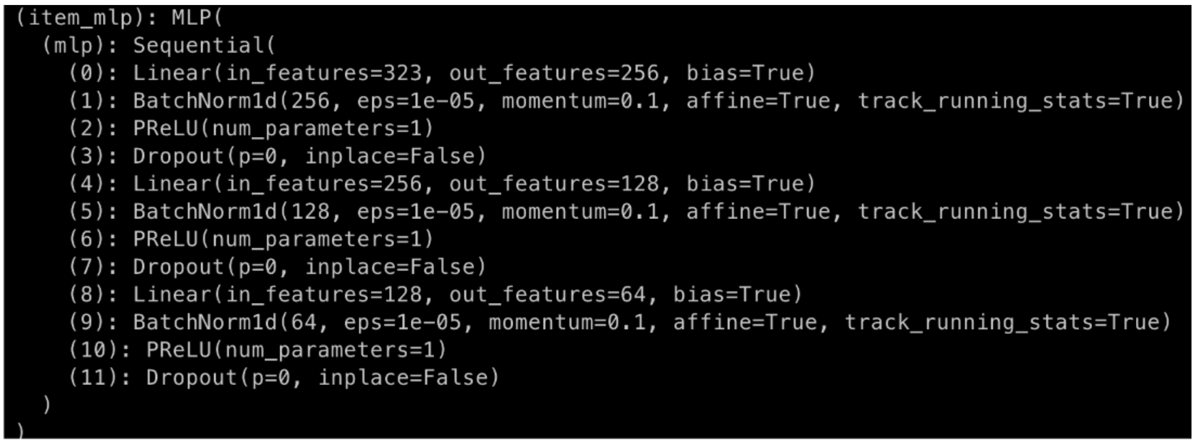
- Feature embedding size: 64
- DNN: [256, 128, 64]
- Activation function: prelu
- Temperature: 1
    - divide after calculating cosine score
- Optimizer:
    - Adam
    - lr: 1e-4
    - weight_decay: 1e-6
- Loss function: torch.nn.BCELoss()
- Annoy
    - embedding matching

## User tower



```
 (user_mlp): MLP(
   (mlp): Sequential(
     (0): Linear(in_features=448, out_features=256, bias=True)
     (1): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
     (2): PReLU(num_parameters=1)
     (3): Dropout(p=0, inplace=False)
     (4): Linear(in_features=256, out_features=128, bias=True)
     (5): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
     (6): PReLU(num_parameters=1)
     (7): Dropout(p=0, inplace=False)
     (8): Linear(in_features=128, out_features=64, bias=True)
     (9): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
     (10): PReLU(num_parameters=1)
     (11): Dropout(p=0, inplace=False)
   )
 )
```

## Course tower



```
(item_mlp): MLP(
  (mlp): Sequential(
    (0): Linear(in_features=323, out_features=256, bias=True)
    (1): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): PReLU(num_parameters=1)
    (3): Dropout(p=0, inplace=False)
    (4): Linear(in_features=256, out_features=128, bias=True)
    (5): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): PReLU(num_parameters=1)
    (7): Dropout(p=0, inplace=False)
    (8): Linear(in_features=128, out_features=64, bias=True)
    (9): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): PReLU(num_parameters=1)
    (11): Dropout(p=0, inplace=False)
  )
)
```

## Post-Processing

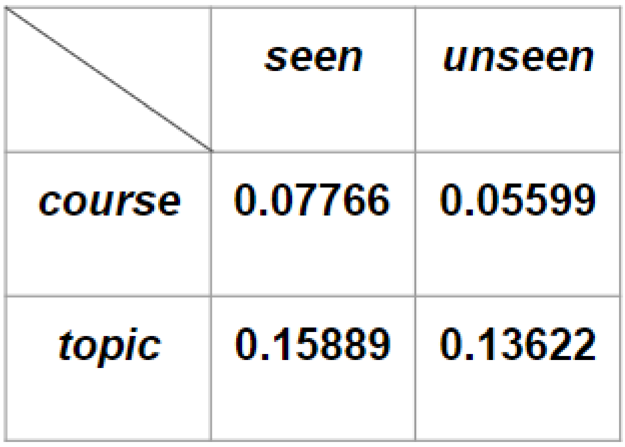
- In the topic prediction, we use the result of the course prediction, change course_id to corresponding subgroup.
- The number of subgroup is usually less than 50.
- We use the aforementioned sampling method to generate subgroups, append until 50.

# Experiment

## Model Performance and model comparison

In this section, we will give some solutions that we tried before, and also give the performance of the model that we finally chose according to the perform.ance of those solution

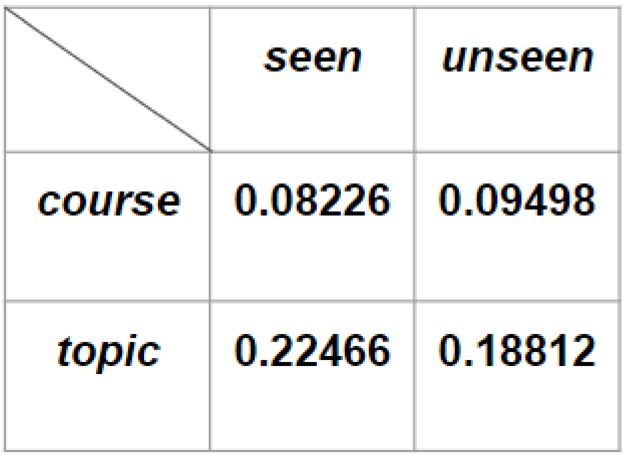
- **Three Solution we tried in this task**
    - **Sample&publish time over price :**
      By analysing the  distribution of Sample&publish time over price between train and validation set, we design a distribution probability function to predict the behavior of test dataset in Sample&publish time over price, and the result shows that even only use obersation approach, the result  isn't that worse than we think.
      ***Public score from kaggle:***



| | seen | unseen |
|---|---|---|
| course | 0.07766 | 0.05599 |
| topic | 0.15889 | 0.13622 |

    - **DSSM :**
      The DSSM model is a two tower DNN model  designed for the recommend system as we preious discussed, and the performance is also the highest among those solutions, so we finally choose it as our  model.
      ***Public score from kaggle:***

✓



| | seen | unseen |
|---|---|---|
| course | 0.08226 | 0.09498 |
| topic | 0.22466 | 0.18812 |

- **Seen(ALS)& Unseen  (information retrival) by TA :**
  This is the solution recommand by TA, a machine learning method that by calculating the probability of the appearcence of the words to predict the best match in course by the query of user's interest. We use almost all information in course.csv except gender, and do the most important extraction in course description, with user's interest, occupation and recreation names, but the result is still not that good as TA provide.
  ***Public score from kaggle:***

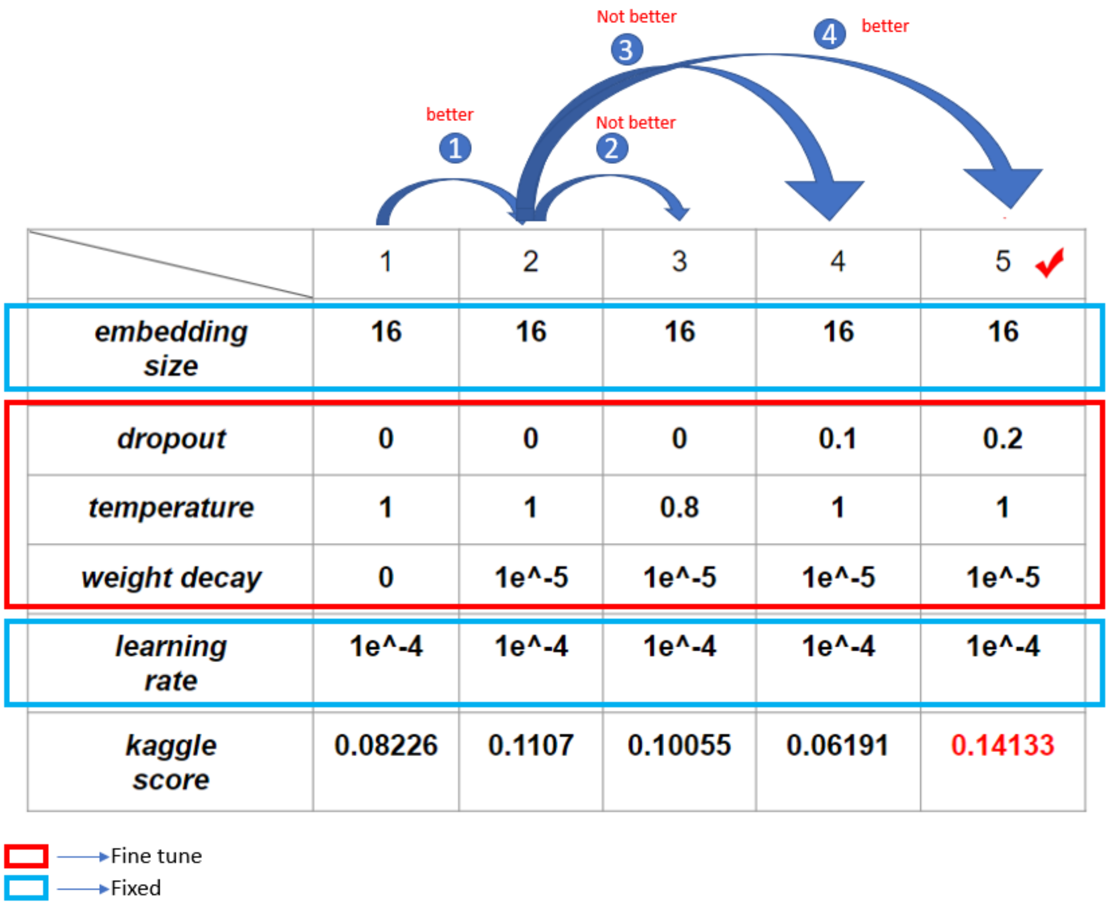
| | *seen* | *unseen* |
|---|---|---|
| *course* | 0.04405 | 0.03624 |
| *topic* | 0.16284 | 0.14149 |

## DSSM Performance

Here we will provide the parameter we use in each sub task and also give the comparison of each parameter to explain why we choose this it.
**P.S.** The model learn in **epoch 10** with each parameter in each sub task, and we only fine-tune only **one** parameter in each comparison.

### Seen course

We fix learning rate in 1e^-4 and embedding size in 16 since the larger learning rate and embedding size has worse performance according to our experiment.

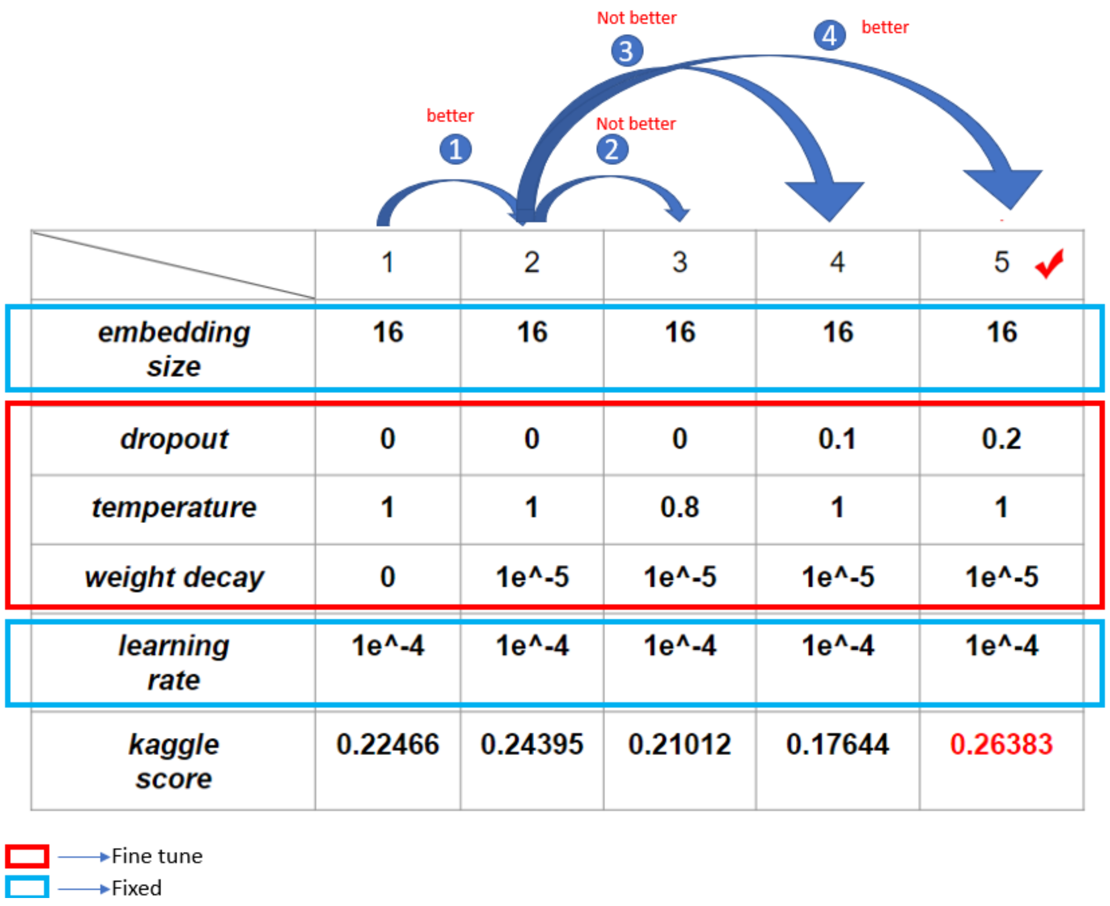

Not better
3
4
better
better
1
Not better
2

| | 1 | 2 | 3 | 4 | 5 ✓ |
|---|---|---|---|---|---|
| *embedding size* | 16 | 16 | 16 | 16 | 16 |
| *dropout* | 0 | 0 | 0 | 0.1 | 0.2 |
| *temperature* | 1 | 1 | 0.8 | 1 | 1 |
| *weight decay* | 0 | 1e^-5 | 1e^-5 | 1e^-5 | 1e^-5 |
| *learning rate* | 1e^-4 | 1e^-4 | 1e^-4 | 1e^-4 | 1e^-4 |
| *kaggle score* | 0.08226 | 0.1107 | 0.10055 | 0.06191 | 0.14133 |

| | |
|---|---|
| 🟥 | Fine tune |
| 🟦 | Fixed |

## Seen topic

Parameter same as seen course.

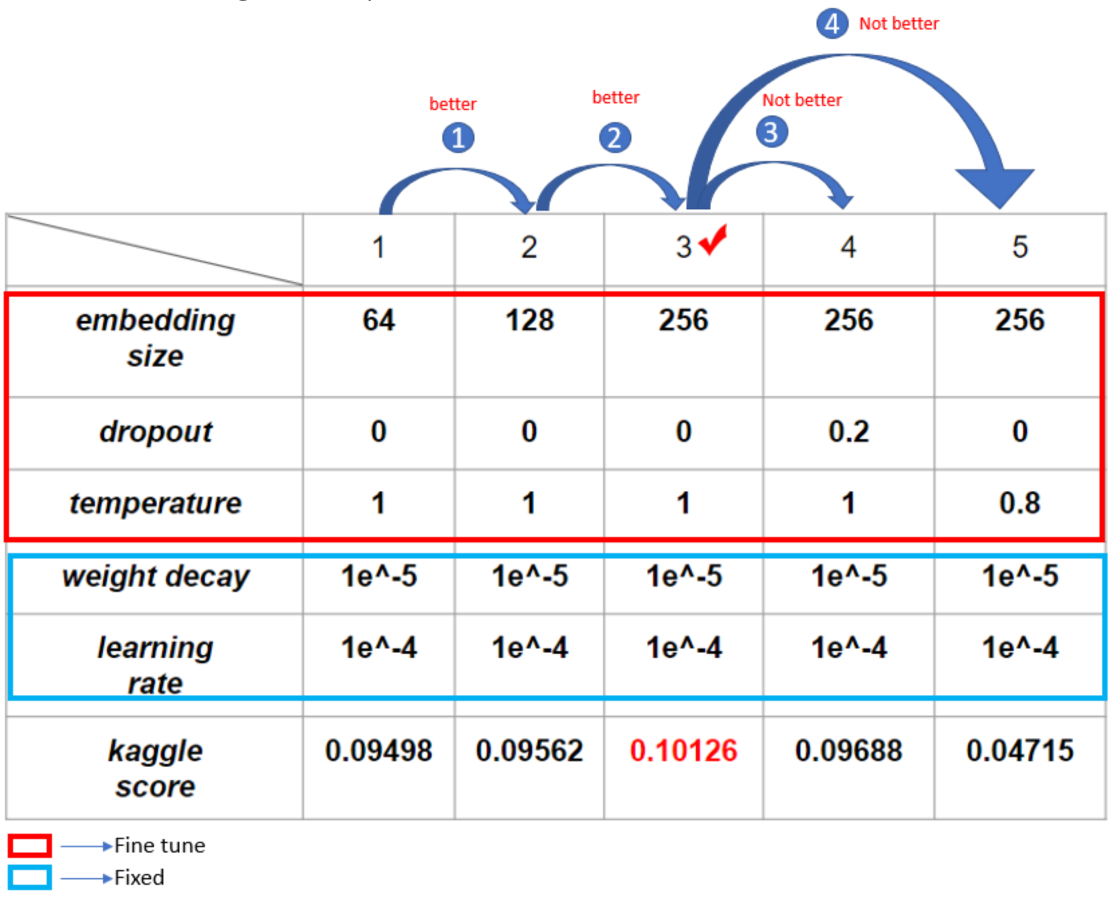

better
1
Not better
2
Not better
3
better
4

| | 1 | 2 | 3 | 4 | 5 ✓ |
|---|---|---|---|---|---|
| ***embedding size*** | **16** | **16** | **16** | **16** | **16** |
| ***dropout*** | **0** | **0** | **0** | **0.1** | **0.2** |
| ***temperature*** | **1** | **1** | **0.8** | **1** | **1** |
| ***weight decay*** | **0** | **1e^-5** | **1e^-5** | **1e^-5** | **1e^-5** |
| ***learning rate*** | **1e^-4** | **1e^-4** | **1e^-4** | **1e^-4** | **1e^-4** |
| ***kaggle score*** | **0.22466** | **0.24395** | **0.21012** | **0.17644** | **0.26383** |

🟥 → Fine tune
🟦 → Fixed

## Unseen course

We fix learning rate in 1e^-4 and weight decay in 1e^-5  since the larger learning rate and no eight decay has worse performance, and in this task, and the larger embedding size has better performance according to our experiment.



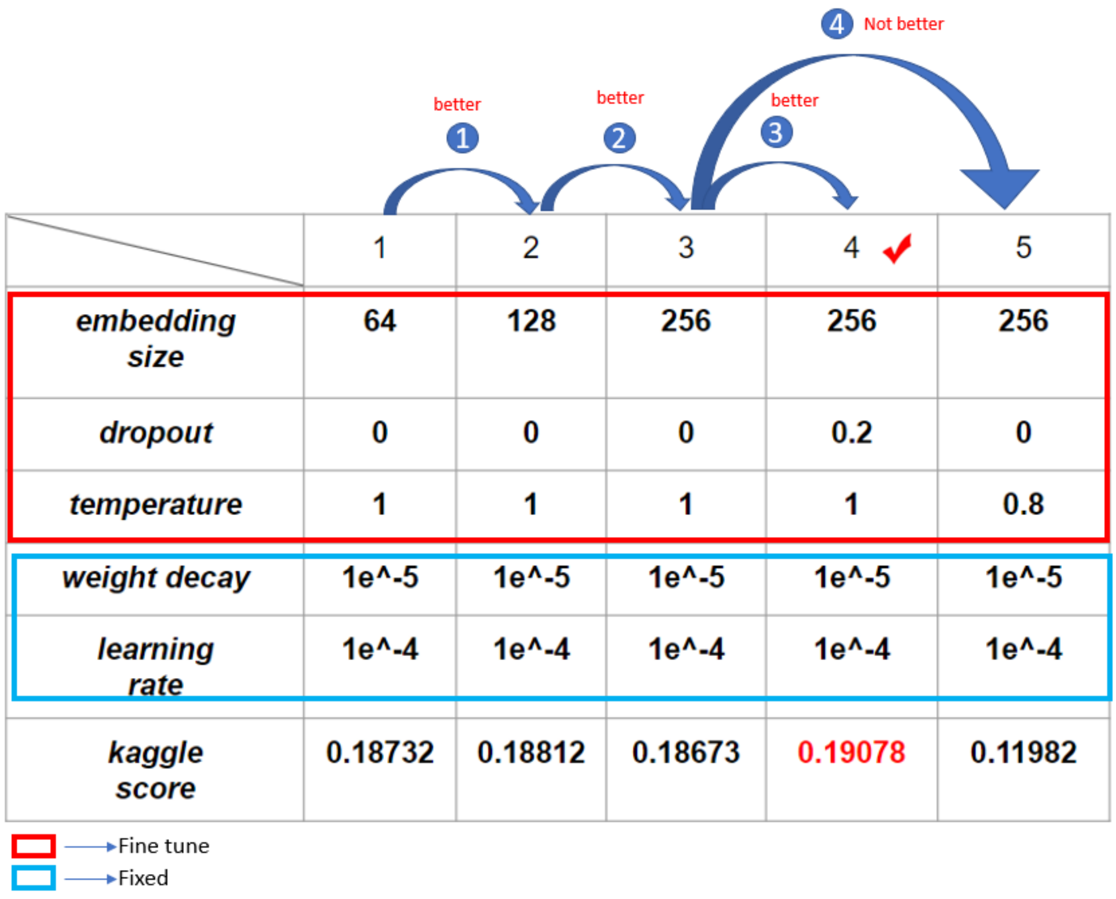
① better
② better
③ Not better
④ Not better

| | 1 | 2 | 3 ✓ | 4 | 5 |
|---|---|---|---|---|---|
| ***embedding size*** | **64** | **128** | **256** | **256** | **256** |
| ***dropout*** | **0** | **0** | **0** | **0.2** | **0** |
| ***temperature*** | **1** | **1** | **1** | **1** | **0.8** |
| ***weight decay*** | **1e^-5** | **1e^-5** | **1e^-5** | **1e^-5** | **1e^-5** |
| ***learning rate*** | **1e^-4** | **1e^-4** | **1e^-4** | **1e^-4** | **1e^-4** |
| ***kaggle score*** | **0.09498** | **0.09562** | **0.10126** | **0.09688** | **0.04715** |

🟥 → Fine tune
🟦 → Fixed

## Unseen topic

Parameter same as unseen course.



④ Not better
better ①
better ②
better ③

| | 1 | 2 | 3 | 4 ✓ | 5 |
|---|---|---|---|---|---|
| embedding size | 64 | 128 | 256 | 256 | 256 |
| dropout | 0 | 0 | 0 | 0.2 | 0 |
| temperature | 1 | 1 | 1 | 1 | 0.8 |
| weight decay | 1e^-5 | 1e^-5 | 1e^-5 | 1e^-5 | 1e^-5 |
| learning rate | 1e^-4 | 1e^-4 | 1e^-4 | 1e^-4 | 1e^-4 |
| kaggle score | 0.18732 | 0.18812 | 0.18673 | 0.19078 | 0.11982 |

🟥 → Fine tune
🟦 → Fixed

## Discussion

QA Some from some other students:

- 1. Q:講者結論中有提到不同GPU可能對結果有很大影響，請問後來有總結出如何判斷當前模型是否有受到GPU影響、以及如何選擇正確的GPU環境嗎?
  A:我們有實測在不同的gpu上做(單張gtx 1080,兩張 rtx 3060以及 單張rtx 3090),三個結果都不同,而最好結果是在rtx 3090上

  至於影響我們認為是因為每張gpu能夠容納的batch size不同,而且batch norm在不同的gpu不會share,才會導致結果差異.
- 2. Q:你好，想請問publish time在2021的training set和validation set長得不太一樣，為何最後還是決定用publish time這個feature呢？
  A:在 Weighted Sampling 的實作上,
  我們沒有直接採用 “2021” 年的 training set 的機率分布.
  而是 “2016~2020” 的部分, 做出 “購買紀錄中該課程的 publish 月份 = x” 的機率 P_m(x)最後再乘以遞增函數 F(year) 來模擬 validation set 的分布.
  Course Prediction Baseline:
  w_k = P_$( price_k ) * P_m( publish month ) * F( publish year )

## Conclusion

**In this task, what we have learned are...**

- The importance of data analysis
    - Data analysis is the first foremost step of DL and ML, in this task, we demonstrate that even if you can't train any model, you still have chance to give a "not bad" prediction in the task by observing the ditribution of sample and price over time or even other relative dta distribution.
- Deciding which ways to solve the problems
    - Each task has differen solutions, by comparing those different solutions, you may be able to choose a better solutions and realize the task is suitable for which kind of solution.
- Different result with GPU environment(single-multi)
    - Different gpu can have different batch size, the batchsize will influence the learning result,  also, single-multi gpu will affect the batch norm, which is not shared across gpus, and gradients resulting from each gpu are averaged, which will also lead to the different result. In our task, for example, we train our model in single 1080, double 3060 and single 3090, results are not same in same parameter, our best result is showed in single 3090.
- Parameter choosing(weight-decay&dropout)
    - Realize the disadvantage of the dask and fine-tune the reasonable parameter. In this task, we consider that the matrix of user and course perchasing is really sparse, and the distribution of course perchasing is extremely unbalance, which may lead to overfit during training, thus, by fine-tuning the parameters -Dropout&Weight decay(penalty), two parameters that are useful for solving overfitting, we finally have a better result comparing to the basic parameter.