Dissertation Title: MOOCs Data Mining

Zunaira Zaman - 20250768

Dissertation 2021

DEPEND Erasmus Mundus Joint MSc in Advanced Systems Dependability

Department of Computer Science,
Maynooth University,
Co. Kildare, Ireland.

A dissertation submitted in partial fulfilment
of the requirements for the
Erasmus Mundus MSc Dependable Software Systems

Head of Department: **Dr Joseph Timoney**
Supervisor: **Dr Guangyuan Piao**
Date: June 22, 2021

Word Count: 14019

Contents

Declaration

I hereby certify that this material, which I now submit for assessment on the program of study as part of Erasmus Mundus Joint Msc in Advanced Systems Dependability (DEPEND) qualification, is *entirely* my own work and has not been taken from the work of others - save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: Zunaira Zaman Date: June 22, 2021

Abstract

Massive Open Online Courses (MOOCs) are attracting unprecedented interest and have been growing rapidly around the world during the coronavirus lockdown. In 2020, enrolments in Coursera were reported to be 640% higher, growing from 1.6 to 10.3 million. However, MOOCs courses have been criticized for the low completion ratio and high number of dropouts. To counter this, researchers have built machine learning algorithms to analyze student data and predict the level of dropout. However, to date, these algorithms have only achieved an accuracy rate of 80%. Our aim is to investigate the dropout prediction problem. Additionally, we will also propose that the learning sophisticated feature interactions behind the user learning activity are critical to maximizing Click-through rate (CTR), and thus improving user retention in MOOCs. A dataset from XuetangX, one of the largest MOOCs in China, was acquired, and a systematic study for dropouts' problems in MOOCs was conducted. We investigated two variants of the CFIN Model with a personalized attention mechanism to model and predict user dropout behaviors. The core of the approach is a course representation model and a user representation model. The CFIN model utilizes the context smoothing technique to smooth feature values with different contexts and use an attention mechanism to combine user and course information into the modeling framework. In addition, the same course can have different dropout rates for different user groups, it was proposed to build a personalized attention network that exploits the embedding of a user ID and embedding of course ID to generate the query vector for the course and learning-activity level attentions. Furthermore, I will also investigate the DeepFM model, which combines the power of factorization machines for recommendation and that of deep learning for feature learning in a new neural network architecture but has not yet been explored for the dropout prediction problem. Experiments on the dataset show that the two proposed variants have similar performance with the CFIN model, predicting dropouts with a 93% accuracy, and it achieves better performance than SVM, RF, and ensemble modeling techniques. Furthermore, then conducted comprehensive experiments to demonstrate the effectiveness and efficiency of the DeepFM model in user CTR for dropout prediction problems. Overall, CFIN model approach has outperformed in performed deepFM, ensemble learning and baseline models with 86% AUC score and F1-score as 93%.

Chapter 1: Introduction

Massive Open Online Courses (MOOCs) are attracting unprecedented interest and growing rapidly around the world during the coronavirus lockdown. By 2020, the number of online students' enrolments has skyrocketed. MOOCs have seen a surge in enrolments since March. In 2020, enrolments in Coursera were reported as 640% higher from mid-March to mid-April than during the same period last year, growing from 1.6 to 10.3 million (Chris, 2020). For another MOOC provider Udemy, enrolments were over 400% between February and March.

However, MOOCs courses are criticised for the low completion ratio and high number of dropouts. Jordan reported that the average completion rate for MOOCs is around 15 percent. EdX, a popular MOOC provider, shows the average completion rate as 5% only. (Feng, 2019) conducted similar analysis for XuetangX dataset, the largest MOOC platform in China, and reported that average completion rate is 4.5% for 700 courses.

Researchers have proposed different methodologies for student's dropout prediction problem ranging from baseline models such as Support Vector Machines (SVM), Logistic Regression (LR), Decision Tree (DT), Random Forest (RF) to neural networks. (E. Mulyani, 2019) have suggested ensemble modelling technique for dropout prediction method and emphasized to use ensemble major voting approach over existing simple baseline models to obtain higher accuracy results. Additionally, one major contribution from google is Wide and Deep neural network model proposed to obtain optimal results for clickstream dataset, which has a huge importance for recommender and prediction systems. Furthermore, (Huifeng Guo, 2017) have proposed the DeepFM method for clickthrough-rate (CTR) predictions. Most importantly, (Feng, 2019) have proposed a context feature interaction network (CFIN) method that is intended to combine context information such as user and course with user learning activity with attention mechanism technique.

In this study, our aim is to find the optimal methodology to identify and predict students who are likely dropout from the course and to improve student's retention in MOOC platforms. After doing systematic study of (Feng, 2019) , we have proposed changes in the original CFIN model. Firstly, to separate the user and course context information in the original CFIN model. Secondly, have performed hypothesis analysis on how user-context features such as education, age, gender etc. have a high correlation impact on student's user's learning activity. For instance, students at undergraduate and graduate level have the highest course video watch time, thus students at undergraduate and graduate level studies with maximum video watch time are less likely to dropout from a course. While students having PhD and associate degrees are more likely to dropout, interesting fact behind this analogy is discovered that user's age effects their learning activity and students in their 20's are more actively participating in course learning activity as comparatively students in their 30's and 40's age.

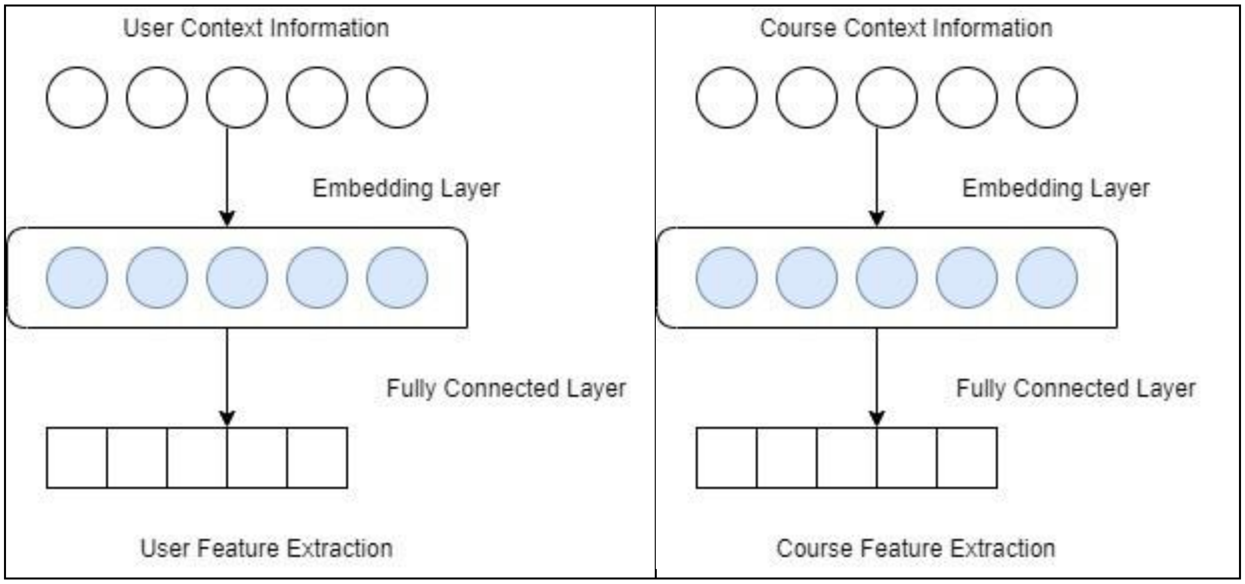
Our main contributions in this study are summarized as follows:

1. Proposed changes in original (Feng, 2019) Context Aware Feature Interaction Network Model for dropout prediction problem.
2. DeepFM methodology applied on MOOC's dataset for learning user low-level and high-level feature interactions for dropout predictions based on CTR prediction.
3. Compared different baseline models and proposed Ensemble Learning solution over existing state-of-the-art prediction methodologies.

1.1 CFIN Modelling Algorithm:

Let us have a brief introduction of CFIN modelling approach being proposed in this paper. So, we have proposed changes in context aware feature interaction network model originally from (Feng, 2019). The core of our approach is user representation module, course representation module, user learning activity module and attention mechanism module. Unlike (Feng, 2019) approach, we have separated the user and course contextual information into separate modules with personalised attention mechanisms to assess the user/course contextual feature interactions impacts both separately and inclusively on user's dropout/retention rates in MOOCs learning activity. With attention mechanism, the model can distinguish the impact of different user/course contexts on user learning activity. Thus, we obtain the final prediction results with the improved matrix factorization method using the user and course contextual feature matrix.
To sum up our proposed changes in the (Feng, 2019) CFIN model approach are as follows:

- We have proposed modelling user and course contextual information separately, which will consider the user encoder and course encoder modules contextual information with personalised attention mechanisms and improve the predictions for dropout or retention.
- Personalized attention mechanism for contextual information helps us to distinguish the effects of user and course context on user learning activity. Which provides more detailed hypothesis results of finding implicit features importance such as user education and age feature interactions with user video watch time.
- We have evaluated the CFIN model on XuetangX dataset which shows consistent improvement and getting results efficiently within less time as comparatively to (Feng, 2019)



1.2 DeepFM Algorithm:

In this paper, the DeepFM algorithm is another notable contribution for student's dropout prediction methods. Despite the popularity of the DeepFM approach in low-to-high feature interactions for clickthrough-rate (CTR) prediction, it is not yet had been applied for MOOCs dropout prediction systems. Students Dropout prediction problem has clickstream dataset, in which it keeps tracks of user learning activity using system tracking log files and regarded as a CTR prediction problem. For CTR prediction

problems, relationships between features and feature interactions and learning implicit feature interactions has a huge impact on predicting accurate results for dropout or retention.
Existing approaches used for CTR prediction problems such as Feed Forward Neural Networks (FFN), and Deep, PNN and Deep and Wide network proposed by Google provides good prediction results but some of those methodologies requires pre-training, and feature engineering. For instance, FFN model only takes high-order feature interactions and requires feature engineering while PPN modeling approach does not require pre-training, but its demerit is that it only considers the low-order features. On the other side, Wide and Deep model does provide both low-order and high-order features, but it requires feature engineering. Thus, the DeepFM is the ideal model for performing both low and high order feature interactions and it does not require pre-training of dataset. Additionally, what makes it more idealist model for CTR prediction that is it does not require additional feature engineering. Let us discuss some low-order and high-order feature interactions of user learning activity with its contextual information. Ideally, explicit feature interactions such as course category and user's gender impacts user learning activity as female students are more inclined to dropout from science courses while male students are highly predicted to dropout from art courses. But as CTR predictions are quite complex and contains loads of dataset which is difficult to interpret for implicit feature interactions and needs experts. Thus, most other feature interactions are hidden in the data, there the DeepFM model provides order-1 (high-order feature interactions) and order-2 (low-order feature interactions) which discovers the implicit relationships between features.

Our main contributions for CTR prediction for MOOCs student's dropout or retention are as follows:

- We proposed to use deepCTR architecture model to implement CTR prediction solution by learning order-1 and order-2 feature interactions in user learning activity log files. By using the factorization machine (FM) and deep neural network (DNN), the model performs both order-1 and order-2 feature interactions, respectively without extra feature engineering.
- We have fine-tuned our model by doing parameter tuning to get optimal results.
- For model performance evaluation, logloss and roc curve scores are used which provides roc score as Finally, based on the predicted CTR, the area under the curve (AUC) and Logloss of the DeepFM method are 0.8323 and 0.3915 on XuetangX dataset.

1.3 Ensemble Modelling Technique:

Furthermore, we have also studied ensemble modeling technique and compared its evaluations results with single classifiers such as support vector machines (SVM), random forests (RF), and decision tree (DT) algorithms. After performing analysis with SVM, RF, and DT using single classifier approach and with ensemble learning technique, we agreed to the conclusion that for clickthrough log files, which generally contains huge dataset, encounters data sparsity, model overfitting, imbalance of classes issues. While using the ensemble majority voting technique we can easily get better accuracy results. For evaluation purpose, accuracy metrics has been used which provides SVM with 77%, DT with 82%, RF as 83% accuracy results while with ensemble modeling technique improvement of 84% has been recorded. In a nutshell, simple baseline modeling techniques are not encouraged to apply for CTR prediction methods.
Henceforth, in this study we have summarized results from baseline modelling technique to ensemble modelling approaches, CTR predictions using DeepFM modelling technique for low-to-high order feature interactions and proposed notable changes in the CFIN modelling approach for student's dropout predictions by using the contextual information.

1.4 Performance Metrics:

The experiments for CFIN model will adopt two performance metrics namely Area Under the Curve (AUC) and F1-Score. However, for DeepFM model approach, we will adopt AUC and logloss (logarithmic loss function). The AUC, F1-score and logloss functions are most used performance metrics for prediction methods.

Chapter 2: Literature Review

In the prior studies, researchers' have proposed different methodologies and algorithms for students' dropout's prediction in MOOCs ranging from baseline models to ensemble techniques, and deep neural networks to complex context-aware prediction solutions. (Ahmed A. Mubarak, 2017) in their recent research on MOOCs dropout or retention prediction have discussed baseline methodologies such as logistic regression, support vector machines, decision forests and random forests, but (Ahmed A. Mubarak, 2017) suggests that baseline methodologies are not sufficient for providing accuracy results for time-series dataset in this case user's learning activity in an enrolled course is considered as sequence labelling or time-series prediction problem. Thus, (Ahmed A. Mubarak, 2017) has proposed two methodologies for students' dropouts: Logistic regression model with regularization term and input-output Markov model approach for time-series prediction problems. Further studies on time-series problems, (Yeung, 2015) proposed a solution using recurrent neural network model with long short-term memory (LSTM). According to (Yeung, 2015), dropout prediction is a sequence classification problem and baseline models does not provide optimal results for sequence classification problems. (Yeung, 2015) proposed temporal model to solve the sequence-labelling problem. (Yeung, 2015) used recurrent neural network (RNN) with long short-term memory (LSTM) and their experimental results shows that with temporal methodologies it obtained superior performance, accuracy score of 84%, which clearly outperforms the baseline model approaches. To improve student's retention in MOOCs another experimental study conducted by (Ahmed A. Mubarak, 2017) investigated the importance of feature correlation with student's retention in MOOCs and they used the naïve bayes (NB), RF, LR, and K-nearest neighbor (KNN) algorithms. Their experimental results depict that baseline classifiers gives more weight to imbalanced classes and results in providing biased results. However, logistic regression outperformed all other baseline classifiers based on accuracy and F1-score. However, these prior approaches have some shortcomings such as imbalance classes, biased results, and data sparsity issue handling. Further studies conducted by (B. Hong, 2017) shows that using a 2-layer cascading technique with a combination of three different classifiers such as RF, SVM and LR can improve the prediction results. Their evaluation results shows that the 2-layer cascading technique has achieved better performance results in terms of precision, accuracy and F1 and area under the curve (AUC) scores. However, a 2-layer cascading technique obtains less recall score comparatively with individual single classifiers (B. Hong, 2017) . Another contribution to solve time-series classification prediction problems in MOOCs is proposed by (Zhemin Liu, 2018) where they have applied neural network approach. (Zhemin Liu, 2018) used the Recurrent Neural Network approach with Long Short-Term Memory (LSTM). With RNN-LSTM architecture the model allows to record user's learning status in the next time step based on her previous learning status (Zhemin Liu, 2018). RNN-LSTM model for student's dropout prediction based on user learning activity status obtained 90% accuracy results for dropout prediction. (Zhemin Liu, 2018) improves the RNN-LSTM model accuracy by updating the data with current input data and prediction model does not retain the historical data log activity record, which is why it provides current learning status of students. Furthermore, using SMOTE and Ensemble learning techniques for prediction methods over baseline models has been proposed by (E. Mulyani, 2019). In their study, they have discussed imbalance classes such as dropout as majority class and non-dropout as minority class which creates biasness in evaluation results. For this purpose, they have encouraged to use synthetic-minority oversampling technique combined with ensemble majority voting classifier. To improve classification results, another suggestion made in the paper is to use different classifiers so that model can predict the accurate results by using different prediction methods. (E. Mulyani, 2019) evaluation results shows that the value for the harmonic mean of precision, recall and F1-score has seen improvement of 7.74% comparatively with previous single classifier approaches. The paper has also compared results for ensemble approach using SMOTE technique and without SMOTE technique, but ensemble learning with SMOTE technique has obtained higher improvement rates as compared to using EL without SMOTE.

Additionally, another notable contribution for dropout prediction problem by (Y. Zhang, 2020) has been proposed in which they used the hybrid depth neural network to model and predict student's user learning activity behavior such as dropout or not dropout within 30 days of course. They have used one-hot encoding to convert dataset into a 2-D matrix factorization and used convolution neural network (CNN). At final step, they used gated recurrent unit (GRU) to extract feature interaction relationships in user learning activity and improve the accuracy performance for prediction method.

In prior studies, we have noticed that prediction methods using deep learning outperformed the baseline methods. Similarly, researchers have focused their attention to contextual information importance and attention mechanism techniques impacts on prediction methods performances. (Wu, 2019) has proposed a neural news recommendation system using personalized attention mechanism. In their methodology, they have implemented solution for news recommendation but using their model architectural framework we can implement a solution for dropout's prediction problem. (Wu, 2019) has separated the user and news contextual information into two models and used CNN layer to find hidden feature representations. They used personalized attention mechanism to train model to attend important feature interactions. Thus, using context aware representation with attention mechanism has outperformed the models without attention mechanism. (Wu, 2019) use of personalized attention mechanism has also outperformed the vanilla attention mechanism as it can adjust the attention weights according to user preferences. Hence, using the contextual informativeness with attention mechanism improves performance accuracy of prediction and recommender systems.

Attention mechanism technique has also been used for XuentanX MOOC dataset available at (MOOCCube, n.d.).In (Feng, 2019) paper, they have proposed context-aware feature interactions model with attention mechanism to predict students dropout rates from a course. With feature augmentation technique, they have combined user and course contextual information and used embedding layer with user learning activity. After getting feature matrix, they added convolution layer and then provided attention weights to different features depending on their feature interaction importance. To summarize, their research contribution in MOOCs is they used context smoothing and attention mechanism which makes it one of unique prediction methods which has scored higher accuracy results than several other state-of-the-art methods. Furthermore, on the same KDDCUP dataset another research has been conducted for course recommendation using attention mechanism. In (J. Wang, 2020) paper, they have used CNN with attention mechanism to model user profiles, thus introducing attention-based prediction method for user ratings. Based on the predicted user ratings, the model will suggest user's the relevant courses in the course recommendation lists.

Furthermore, we have also studied low-to-high feature interaction networks for the CTR based prediction methods. In the (Huifeng Guo, 2017), they have proposed a new solution method known as DeepFM, which emphasizes both low-order and high-order feature interactions. The DeepFM model has the power of factorization machines and deep learning for feature engineering. The DeepFM method does not require pre-training which saves the time and is efficient process. The DeepFM method approach also does not require feature engineering as comparatively other methodologies used for CTR prediction such as FNN, PNN and Deep and Wide Model proposed by Google. Despite, its popularity we have observed that the DeepFM method has not yet been applied for MOOCs dropouts prediction. However, we have seen the research studies which have used the DeepFM methodology for e-commerce (J Xu, 2021) , stock prediction (J. Huang, 2019), disease prevention systems (Z. Yu, 2021). In (J Xu, 2021) paper, they have used the DeepFM approach for CTR prediction by doing parameter tuning in the available deepctr package (deepctr.models.deepfm.html, n.d.).

In this research study, we have two goals: firstly, to propose changes in the original CFIN model (wzfhaha, n.d.), we have proposed changes to separate user-context module and course-context module using personalized attention mechanism. To measure the performance prediction results we have used the ROC curve, and F1-score metrics. Additionally, to learn feature interactions importance in MOOCs XuetangX dataset available at (moocdata.cn, n.d.). By employing the DeepFM on MOOCs dataset, we can perform order-1 to order-2 feature interactions efficiently on user learning activity which contains clickstream dataset. With this methodology it is easier to learn implicit feature interactions which will help to make accurate dropout predictions based on user clickthrough-rate in an enrolled course activity. For the DeepFM evaluation performance, we will be using ROC curve, accuracy, and F1-score metrics.

Chapter 3: The CFIN Model for Dropout Prediction in MOOCs

3.1 Context Feature Interaction Network:

In this section, we performed different hypothesis in context aware feature interaction network built and proposed by (Feng, 2019). Meanwhile, we compared the existing CFIN model (Feng, 2019) for student's dropouts prediction problem with other similar models. We found two helpful research studies based on neural network attention mechanism approaches which are being used for case studies such as neural news recommendation with attention mechanism (Wu, 2019) and attention-based CNN for personalized course recommendation (J. Wang, 2020). After doing systematical analysis, we have proposed several changes in the CFIN model. Such as separating the course-context module and user- context module in the CFIN model. In this CFIN model, it will have three modules: context smoothing, attention mechanism module, and context environment overall module. In Context smoothing, it will use feature augmentation, embedding and feature fusion to smooth feature values from different context. While attention mechanism is used to determine the impact of different context information such as user and course context on user learning activity. Additionally, context overall module provides the effect results of overall context environment on the user learning activity and dropout prediction rates. To summarize CFIN model, attention based neural network model will be used to get user profiles and predict student's dropout rates on the courses.

3.2 The Proposed Method:

In the schematic illustration of the model in figure below, the framework has feature extraction, feature augmentation, K-means clustering of objects, and CFIN method which further includes four components such as feature embedding, attention layers, CNN layers, and SoftMax layer. Before proceeding with attention-based CNN model approach, let us first define the modeling algorithmic inputs.

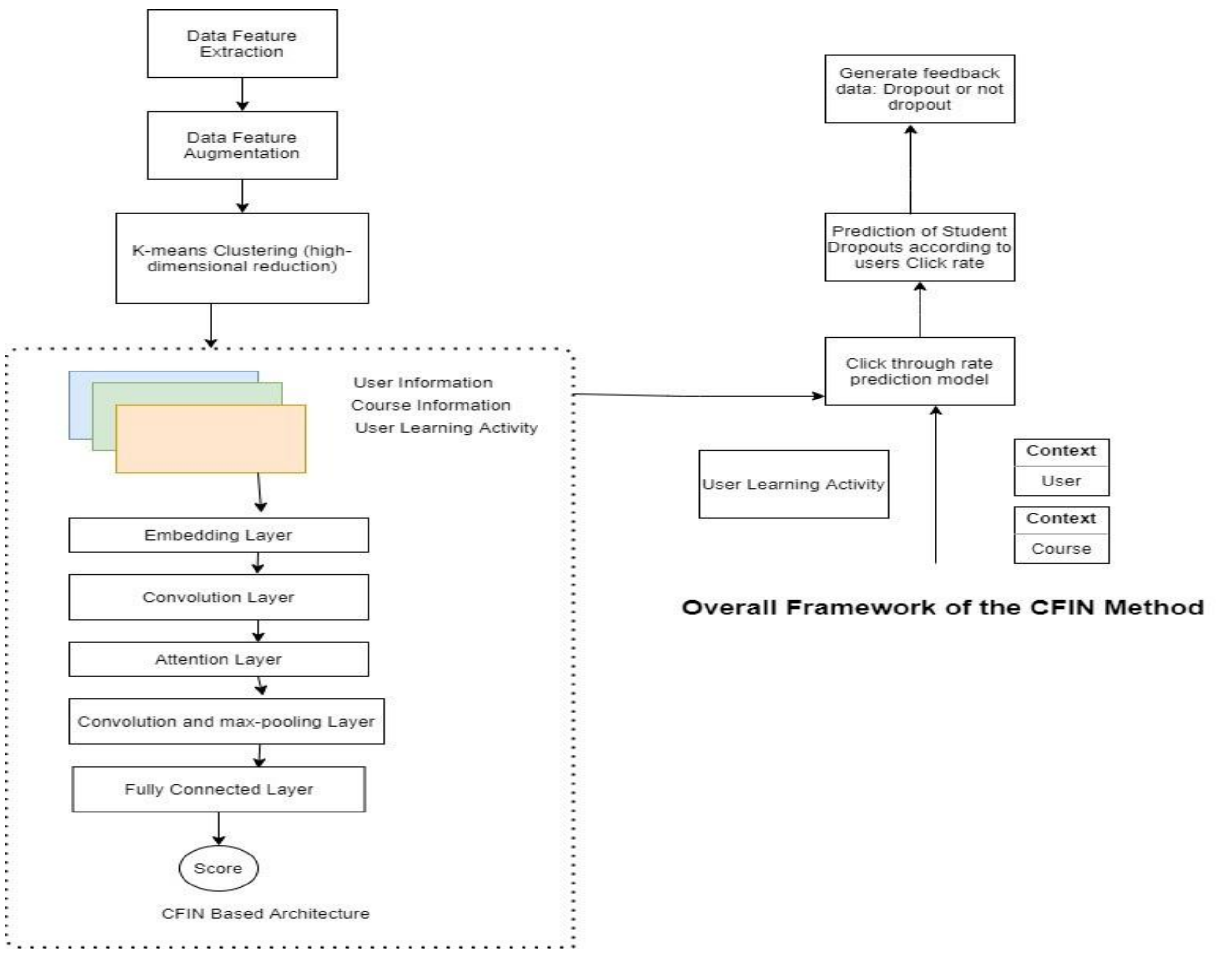
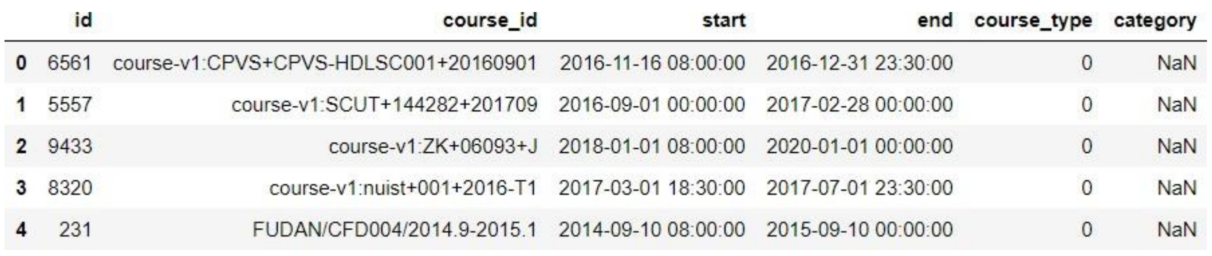

Data Feature Extraction
Data Feature Augmentation
K-means Clustering (high-dimensional reduction)
User Information
Course Information
User Learning Activity
Embedding Layer
Convolution Layer
Attention Layer
Convolution and max-pooling Layer
Fully Connected Layer
Score
CFIN Based Architecture
Generate feedback data: Dropout or not dropout
Prediction of Student Dropouts according to users Click rate
Click through rate prediction model
User Learning Activity
Context
User
Context
Course
Overall Framework of the CFIN Method

Figure 3:0:1 Overall Framework of CFIN Model

Definition 1: Course Context Information

Course context information comprises course type, category and timestamp data of course starting and ending duration time. Course information will be divided into two parts: course information and statistical information. Statistical information is obtained from the tracking log files which contains information of all users enrolled in a course. Such as user video log files, assignment reports etc..



	id	course_id	start	end	course_type	category
0	6561	course-v1:CPVS+CPVS-HDLSC001+20160901	2016-11-16 08:00:00	2016-12-31 23:30:00	0	NaN
1	5557	course-v1:SCUT+144282+201709	2016-09-01 00:00:00	2017-02-28 00:00:00	0	NaN
2	9433	course-v1:ZK+06093+J	2018-01-01 08:00:00	2020-01-01 00:00:00	0	NaN
3	8320	course-v1:nuist+001+2016-T1	2017-03-01 18:30:00	2017-07-01 23:30:00	0	NaN
4	231	FUDAN/CFD004/2014.9-2015.1	2014-09-10 08:00:00	2015-09-10 00:00:00	0	NaN

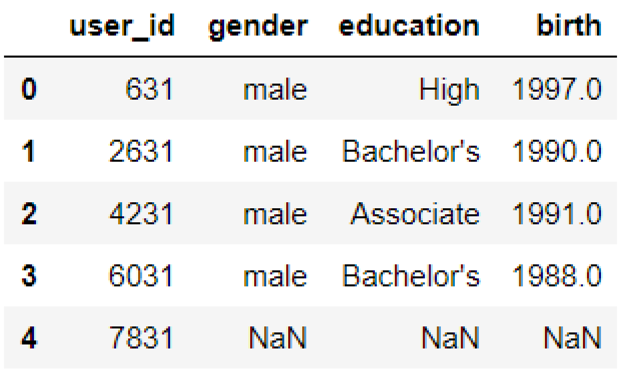
Figure 3:0:2Course Information

Course Context Information		
	Name	**Description**
Course	Category	Course category (math, physics ,electrical, computer, foreign

		language, business, economics, biology, medicine, literature, philosophy, history, social science, art, engineering, education ,environment, chemistry)
	Start Date	Course start timestamp
	End Date	Course end timestamp
	Course Type	Course type (self-paced or
Video	Play	Avg number of users watched Video
	Stop	Avg number of users stopped Video
	Jump	Avg number of users jumped during video
	Seek	Avg seek/view time of users in a Course

<u>Definition 2: User Context Information</u>

User context information has been divided into two categories: user demographics and user online statistical information. User demographics information includes user gender, education level, and their age. While statistical information will include user's online learning behavior in courses such as user forum activities (question/answer), video activities (watch/stop/jump), assignment activities. It will alsoinclude user session information that counts the number of times user logged on and total spend time per day on a course.



	user_id	gender	education	birth
0	631	male	High	1997.0
1	2631	male	Bachelor's	1990.0
2	4231	male	Associate	1991.0
3	6031	male	Bachelor's	1988.0
4	7831	NaN	NaN	NaN

Figure 3:0:3Course Information

In the table, user context information (user demographics and user statistical data) has been displayed. By statistical behavior, it means the user learning behavior statistics such as total number of enrolled courses, total number of dropout courses, average number of videos watched, stopped or jumped etc.. We will represent the users enrolled in a course with below equation:

$C^u \in C$, C represents courses and C^u represents the user enrolled in a course C (J. Wang, 2020)

While users enrolled in a course c will be represented as X(u, c), briefly represents user u enrolled in a course c.

User Context Information		
	Name	Description
User Demographics	Gender	Gender value (0 is unknown, 1 is male and 2 is female)
	Education	User education background (Bachelor, High, Master, Primary,Middle, Associate, Doctorate
	Birth	User's age
Video	Play Video	Average play video time
	Seek Video	Average num of seek video time
	Stop Video	Average num of stop video time
	Pause Video	Average num of pause video Time
	Load Video	Average num of video load time
Course	Enrollments Number	Total num of enrolled courses
	Dropouts Number	Total num of dropout courses
Forum action	Create Question	Avg num of question posted
	Create answer comment	Avg num of questions answers
Assignment	Reset problem	Avg reset ratio
	Problem correct ratio	Avg number of correct ratios
	Problem revised ratio	Avg number of incorrect ratios
Click Action Habit	#Click Courseware	Avg number of clicks
	#Click About	Avg number of about clicks
	#Click Forum	Avg number of forums clicks

Figure 3:4 User Context Information

While user online information will be stored in the system tracking log files. With the hypothesis performed on the dataset, the (J. Wang, 2020) suggests that the most active users on videos, forums and assignments will have the lowest dropout rates probability in MOOCs. In the above statistical analysis user information, it reflects users learning activity in an enrolled course and describes overall CTR features such as counts user online actions, average users click on forum posts, their assignments average correct and incorrect ratios.

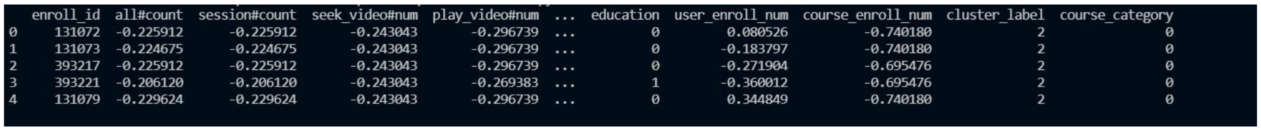
Definition 3: Enrollment Relation

To formulate this problem more precisely, user enrollment in a course has been defined as set of enrollments as follows. Let define user as U, course as C and E(u, c) denotes the user U enrolled in a course C. Thus, E{(u, c)} will represent the set of all users U enrolled in a course C. (Feng, 2019)

Definition 4: User Learning Activity

When a user enrolls in a course on XeutangX MOOC platform, it maintains a user learning activity tracking log file. In the tracking log files, it records user click action data on video activities, assignment activities, forum activities, and webpage activity. According to (Feng, 2019), the learning activity can beformulated as follows:

An m_x dimensional vector X(u,c) where it represents that the user u enrolled in a course c. And X(u, t, c) will represent the continuous feature values associated with user u enrolled in a course c and its learning activity as t.



```
   enroll_id  all#count  session#count  seek_video#num  play_video#num  ...  education  user_enroll_num  course_enroll_num  cluster_label  course_category
0     131072  -0.225912      -0.225912       -0.243043       -0.296739  ...          0         0.080526          -0.740180              2                0
1     131073  -0.224675      -0.224675       -0.243043       -0.296739  ...          0        -0.183797          -0.740180              2                0
2     393217  -0.225912      -0.225912       -0.243043       -0.296739  ...          0        -0.271904          -0.695476              2                0
3     393221  -0.206120      -0.206120       -0.243043       -0.269383  ...          1        -0.360012          -0.695476              2                0
4     131079  -0.229624      -0.229624       -0.243043       -0.296739  ...          0         0.344849          -0.740180              2                0
```

3.4 Data Feature Extraction

The first step of our model is to do feature preprocessing to feed the dataset to CNN layer. The current dataset we have is of raw activity record. CNN only accepts dataset in low-dimensional form; thus, we are required to perform feature preprocessing. The dataset for user information contains different features and those features have different meanings such as userID, birth, education level, gender, and user's learning activity. Each activity will record different information depending on the user's actions. Similarly, for course records information it contains course start time and end time, course category andcourse type. The first goal is to do feature preprocessing for each user and course information. The attributes in user information such as user gender and education can be represented with one-hot vector encoding while the user age is represented with continuous feature variable.

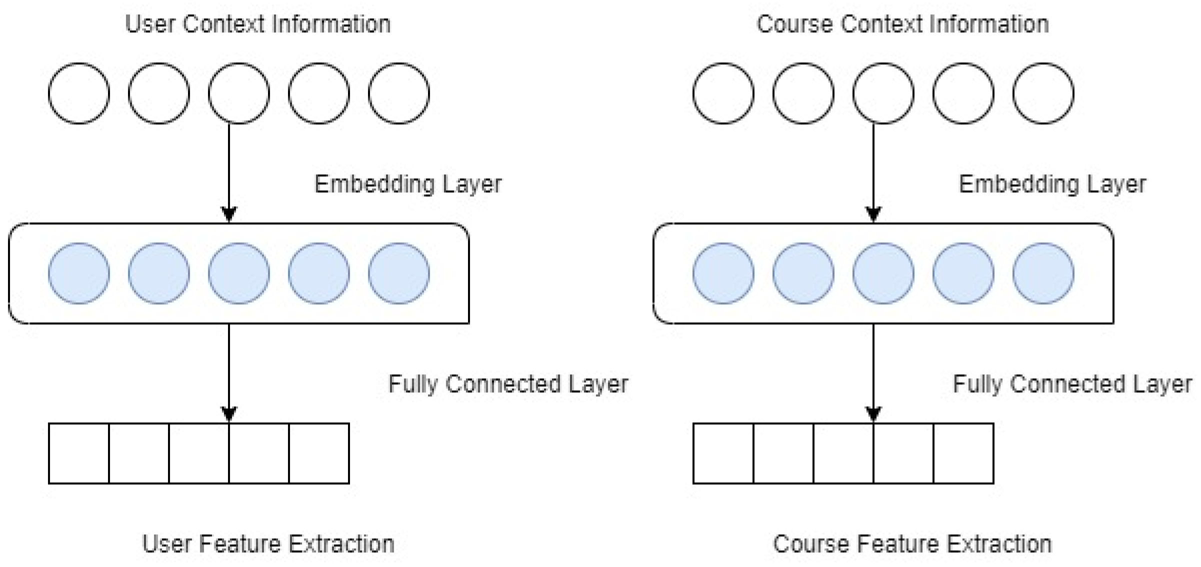


Figure 3:5 Feature Extraction (J. Wang, 2020)

Similarly, for course information course category and course type will be represented using one-hot vector encoding and course start and end time will be converted into timestamp dataset.

As per the above figure, user encoder and course encoder have three-basic layers: Embedding Layer, Convolution Layer, and Personalized Attention.

3.5 Feature Augmentation:

In feature augmentation section, for user encoder its feature learning activity X(u, *) is expanded with its user context statistics.

(Feng, 2019) suggests user context and course context definition as:

User Context: $g_u(a_i) = g_u = A_i(u, c) \rightarrow [avg(\{A_i(u,*)\}), \max(A_i(u,*))]$

Course Context:
$g_c(a_i) = g_c = A_i(u, c) \rightarrow [avg(\{A_i(*, c)\}), \max(A_i(*, c))]$

Users Context equation: u denotes user, c denotes course, A(u, c) represents user u activity in course c.

Augmented Feature Activity:

$$A = A^1{}_g + A^2{}_g + .. + A^m{}_g$$

Where user learning activity augmentation: $A^1_u = [\, x_i + g_u(x_i) \,]$

And course learning activity augmentation: $A^1_c = [\, x_i + g_c(x_i) \,]$

Embedding Layer:

According to the structure of Feature Processing in above user and course context, the model architecture requires to create dense feature vector through embedding layer.

Embedding Equation: (Feng, 2019)

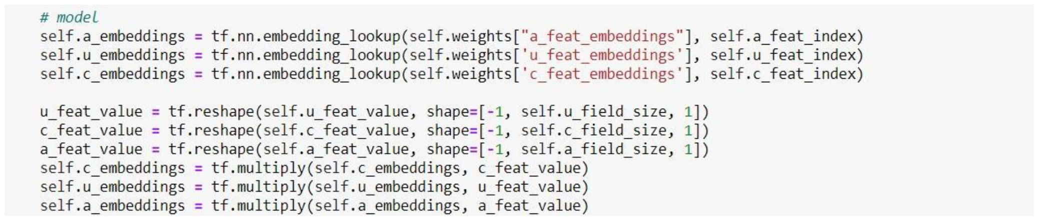
$e\ = \hat{x}a$

Firstly, let's create embeddings for both user and its learning activity which contains dataset of video, forum, assignments, click actions. For this purpose, we used manual coding technique to do feature extraction for user learning activity. Where we will count the averagenumber of user activity on different activities.

video_action = ['seek_video','play_video','pause_video','stop_video','load_video']
problem_action =
['problem_get','problem_check','problem_save','reset_problem','problem_check_correct', 'problem_check_incorrect']

forum_action =
['create_thread','create_comment','delete_thread','delete_comment']
click_action =
['click_info','click_courseware','click_about','click_forum','click_progress']

The model specifies the embedding size = 32, and also specifies the number of output latent vectors in the resulting matrix.
At this step, use the dot product of both user and learning activity embedding using "merge layer".
Coding Example: Note: we used random weights and biases while creating embedding layersusing TensorFlow



```
# model
self.a_embeddings = tf.nn.embedding_lookup(self.weights["a_feat_embeddings"], self.a_feat_index)
self.u_embeddings = tf.nn.embedding_lookup(self.weights['u_feat_embeddings'], self.u_feat_index)
self.c_embeddings = tf.nn.embedding_lookup(self.weights['c_feat_embeddings'], self.c_feat_index)

u_feat_value = tf.reshape(self.u_feat_value, shape=[-1, self.u_field_size, 1])
c_feat_value = tf.reshape(self.c_feat_value, shape=[-1, self.c_field_size, 1])
a_feat_value = tf.reshape(self.a_feat_value, shape=[-1, self.a_field_size, 1])
self.c_embeddings = tf.multiply(self.c_embeddings, c_feat_value)
self.u_embeddings = tf.multiply(self.u_embeddings, u_feat_value)
self.a_embeddings = tf.multiply(self.a_embeddings, a_feat_value)
```

Convolution Layer:

After getting embedding layer, there comes the 1-dimensional convolution layer which proved to be an effective approach for finding local context information. In this study, user and course context are important with learning activity to find implicit features such as user's video activities and assignment activities effecting user's dropout rates in a course. Thus, we applied CNN architecture on user learning activity and course context and user context to learn its implicit feature interactions in user activity such as video, forum, assignment activities implicit effects on user's dropouts.

$$V_g^i = \mathrm{ReLU}(W_{conv}\,\delta(E_g^i + b_{conv}))$$

In this equation (Feng, 2019), E denotes the embedding matrix vector, $\delta(E^i)_g$ represents the flattening matrix for embedding layer, W is convolution kernel, B represents the bias term in the matrix. For CFIN model, a non linear activation function Relu has been chosen. The output of one-dimensional CNN layer will be represented as a Tensor

$$[c_1, c_2, .. c_m]$$

Using Tensorflow, it has been obtained in the following way:

```
self.a_embeddings = tf.nn.conv1d(self.a_embeddings, self.weights['a_conv_filter'],
stride=5, padding='VALID', data_format='NWC') + self.weights['a_conv_bias']
self.a_embeddings = tf.nn.relu(self.a_embeddings)
```

Personalized Attention Layer:

In this study, our goal is to use attention based CNN layer to predict the dropout rate by modeling the attention-based interactions for user learning activity feature with its user and course context.

$$V_z = \sigma(W_{fc}\delta(E_z) + b_{fc})$$

(Feng, 2019) suggests the attention layer equation, where V_z denotes the attention score for model.

$$c_i = \mathrm{Relu}(W\,(E_z)\ +\ b)$$

In the above equation, E_z is the input of convolution layer, b is the bias term and Relu has been used as an activation function.

3.6 Data Description:

XuetangX MOOC platform dataset is used for the student dropout prediction case study using CFIN based approach. Before proceeding with modelling CFIN framework, user clustering analysis over user learning activity is performed using k-means algorithm. The K-means user clustering helps to group users into different categories based on user's studying behavior. Furthermore, hypothesis to find user's major dropout reasons, user retention, and user's friend influential to dropout from a course are conducted.

Some experiments on the user-context information shows that the user gender, education, and age features have a decent influence on user dropout probability rate in a course. For instance, (J. Wang, 2020) suggests that users with highest number of video watch are at lowest risk of dropouts. In the figure below, users' gender has been represented as unknown with 0, 1 as male, 2 as female.

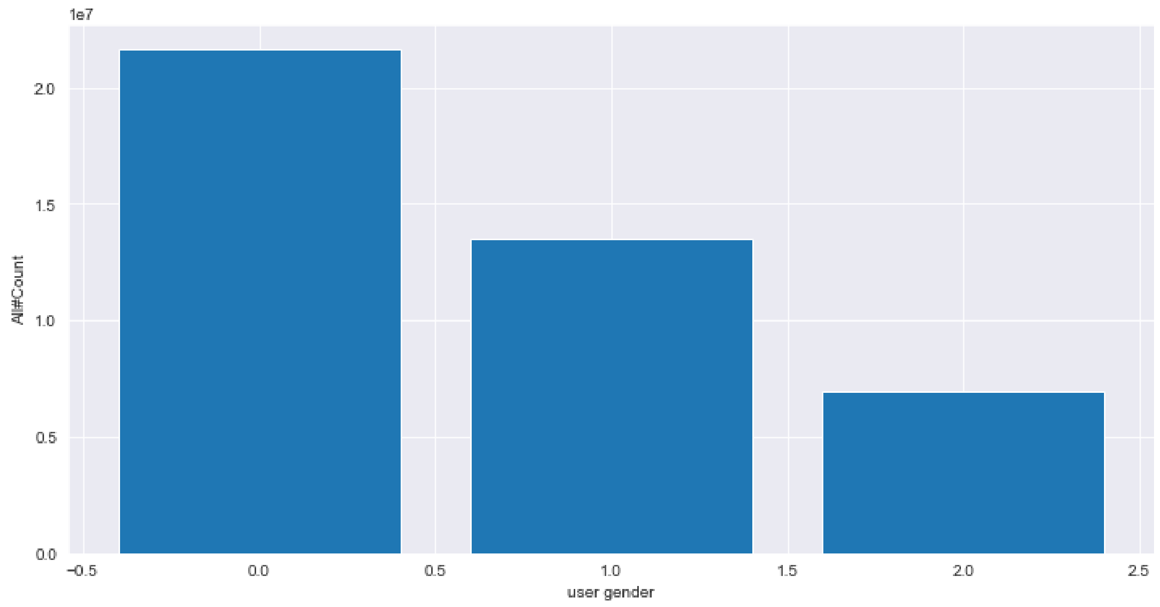


Figure 3:6 User Gender and dropouts probability rates

In the figure below, it displays how user's gender has an overall effect in course video activities (play, pause, stop, seek, load videos in a course)

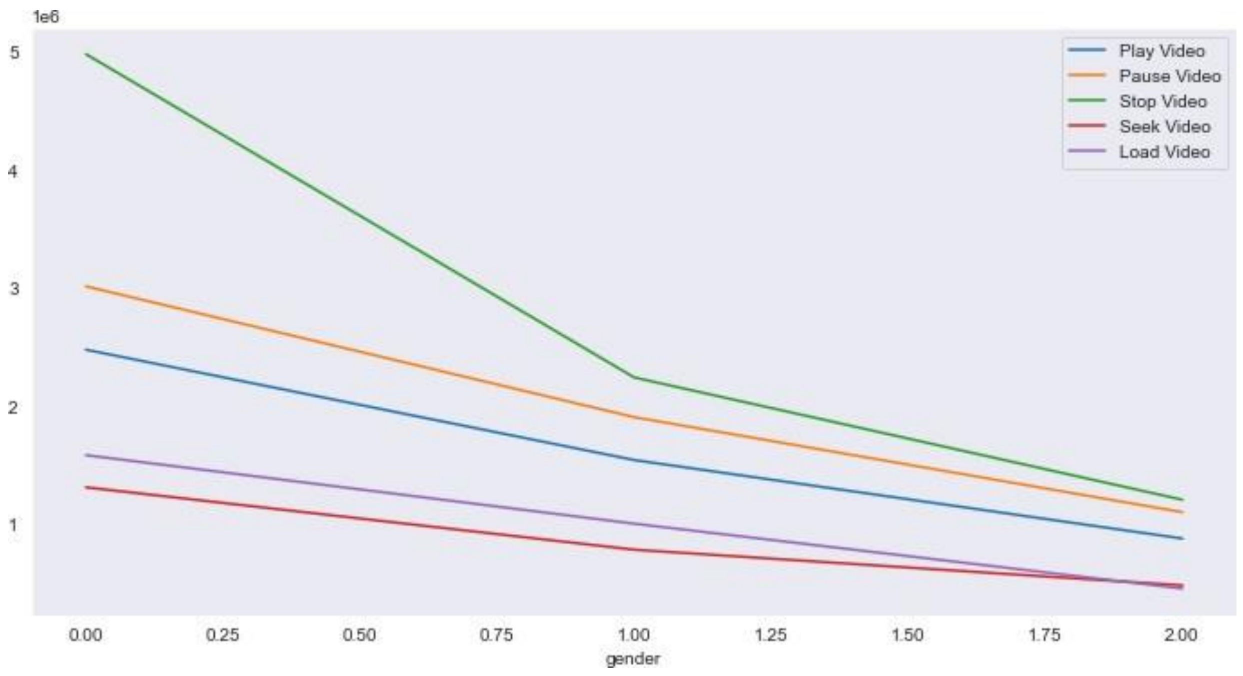


Figure 3:7 User Gender effects on Course Video Activities

In this figure, user education level has been used as a group by query and its impact on vide activity has been visualized.

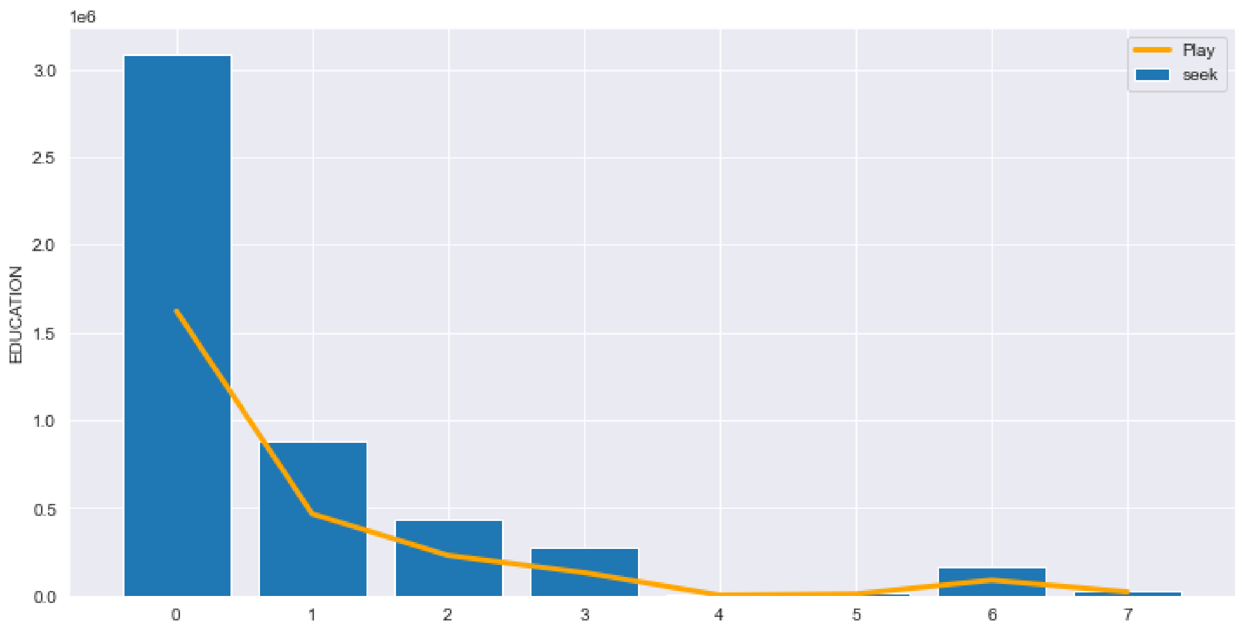


Figure 3:8 User Education impact on Video Activity (Play, Seek)

Furthermore, we found out that user's education level also influences their dropout probability in a course. User's coming from different educational backgrounds such as Bachelor's , High, Master's, Primary, Middle, Associate, Doctorate impacts their dropout rates. For instance,

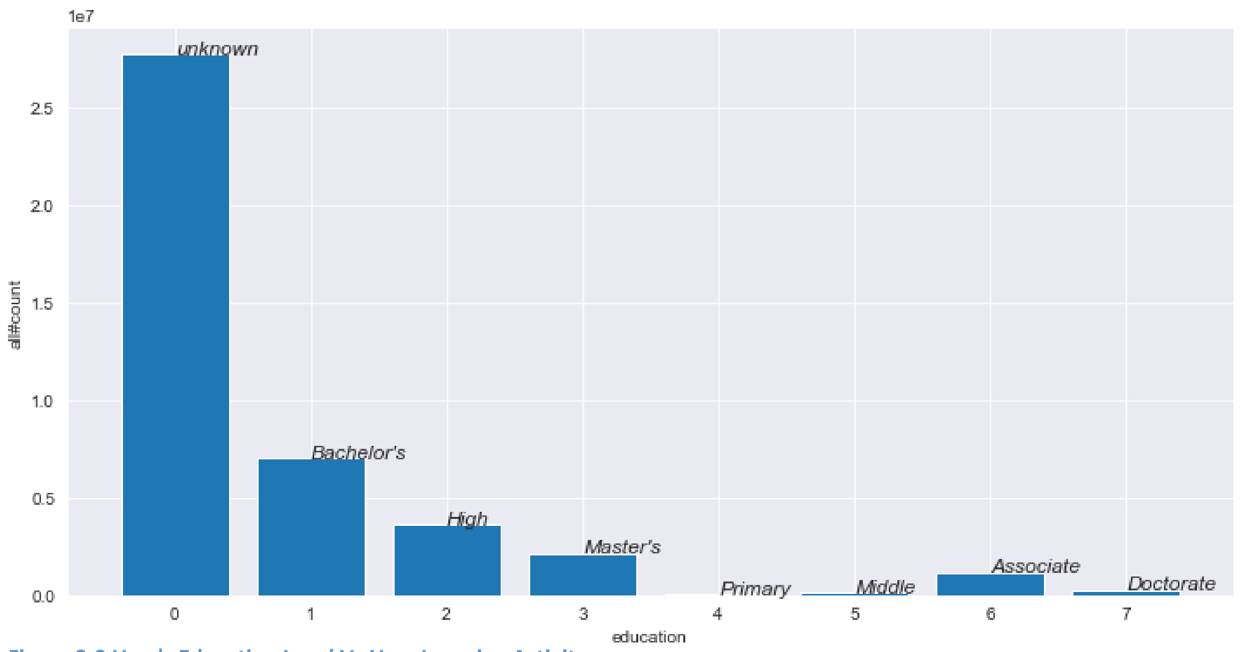
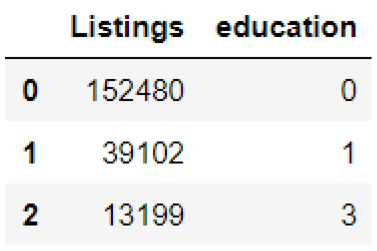


Figure 3:9 User's Education Level Vs User Learning Activity

In the figure, user's with bachelor's, High School, and Masters education level have the lowest dropout rates. While dropout rates are higher for Associate and Doctorate education level. With this analysis, it also represents that user's age also affects the student dropout rates.

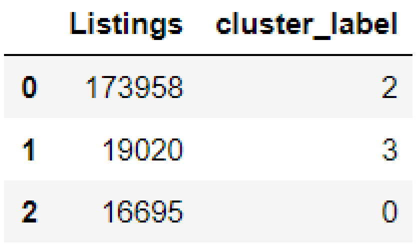
Similarly, to find out the maximum user learning activity from the perspective of user cluster label and user education. In the table below, maximum user learning activity grouped by user education has beendisplay where bachelor denoted by 1, Masters denoted by 3 and the value for unknown education level is said to be 0.



	Listings	education
0	152480	0
1	39102	1
2	13199	3

Figure 6Top 3 Education Categories with Maximum Learning Activity

In the table below, it takes the user cluster dataset which are total 5 user clusters. It shows that users in cluster label 2 are more hardworking and have the longest 10X greater learning rate as compared to other clusters. And ranked user cluster 2, 3, and 0 as the user clusters with maximum learning activity.



	Listings	cluster_label
0	173958	2
1	19020	3
2	16695	0

Figure 7Top Three User Clusters with Maximum Learning Activity

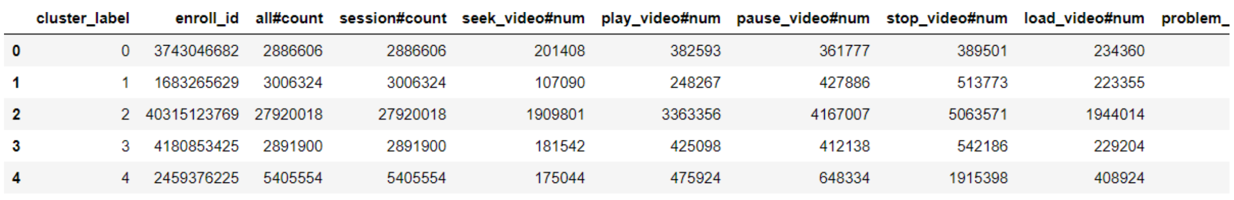
3.6.1 User Clustering using K-means:

Clustering is an efficient approach to reduce data dimensionality and to reduce data sparsity and model overfitting problems. The dataset contains different types of data such as user, course and learning activity dataset. There are some complex implicit relationships between user and their probability to dropout from a course. With K-means algorithm, it reduces the dimensionality from the perspective of similarity in users, courses, and learning activity click log dataset. (Feng, 2019)

The number of clusters are set to k=5, for a user u enrolled in a course c is defined as a binary valued vector, such as and S^u_c represents if a user u will visit the course c.

$$S^u_c = \{ S^u_{c,1} , S^u_{c,2}, .. , S^u_{c,k} \}$$

This clustering analysis results are displayed in the below table figure. The dataset has been divided into five clusters. The cluster 2 and cluster 5 have the lowest dropout rates and the users of cluster 2 have the longest video watching time and have the maximum number of questions posted on forums which is10X higher than other user clusters.



	cluster_label	enroll_id	all#count	session#count	seek_video#num	play_video#num	pause_video#num	stop_video#num	load_video#num	problem_
0	0	3743046682	2886606	2886606	201408	382593	361777	389501	234360	
1	1	1683265629	3006324	3006324	107090	248267	427886	513773	223355	
2	2	40315123769	27920018	27920018	1909801	3363356	4167007	5063571	1944014	
3	3	4180853425	2891900	2891900	181542	425098	412138	542186	229204	
4	4	2459376225	5405554	5405554	175044	475924	648334	1915398	408924	

Figure 9Results of Clustering Analysis

In the bar chart figure, we have displayed the user overall learning activity count. Thus, as user cluster 2 and 5 are most hardworking and it is predicted that these will have the lowest dropout rates.

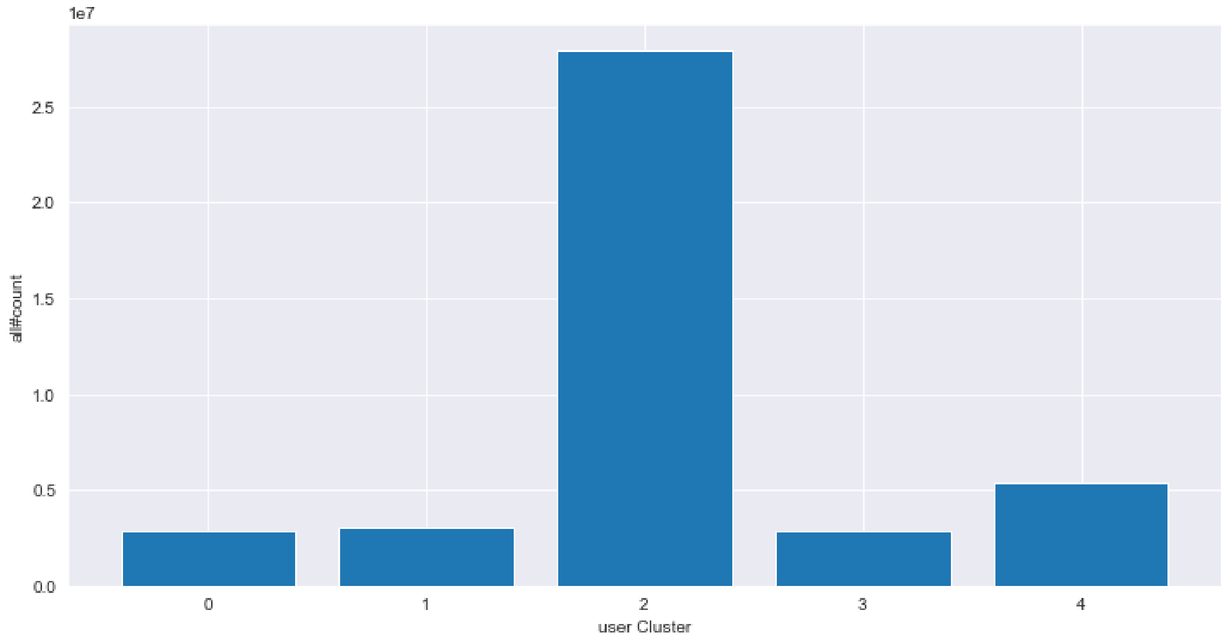


Figure 0:4 User Clustering Vs User Learning Activity

3.7 CFIN Framework Approach

We have proposed to separate the course context and user context module and later embed the modules in attention-based CNN architecture. There are three major modules: User Encoder, Course Encoder, and Click Predictor. In the user encoder module, it aims to learn user profile representation based on the number of times the user clicked on a course activity such as watch/stop/jump course video, forum activities. In the course encoder, the goal is to learn course representation. While in the Click Predictor module, it is used to estimate the click through rate of a user learning activity. In the model architecture figure, it has been displayed that we are using attention-based CNN mechanism for user and course modules to predict student's dropout rates in MOOCs. In this model, CNN with attention mechanism is used in course-context and user-context to extract the course and user features.Additionally, the strength of using attention mechanism in CNN model is to find latent factors. Finally, the proposed model will obtain latent features both at personal and statistical levels. (J. Wang, 2020)

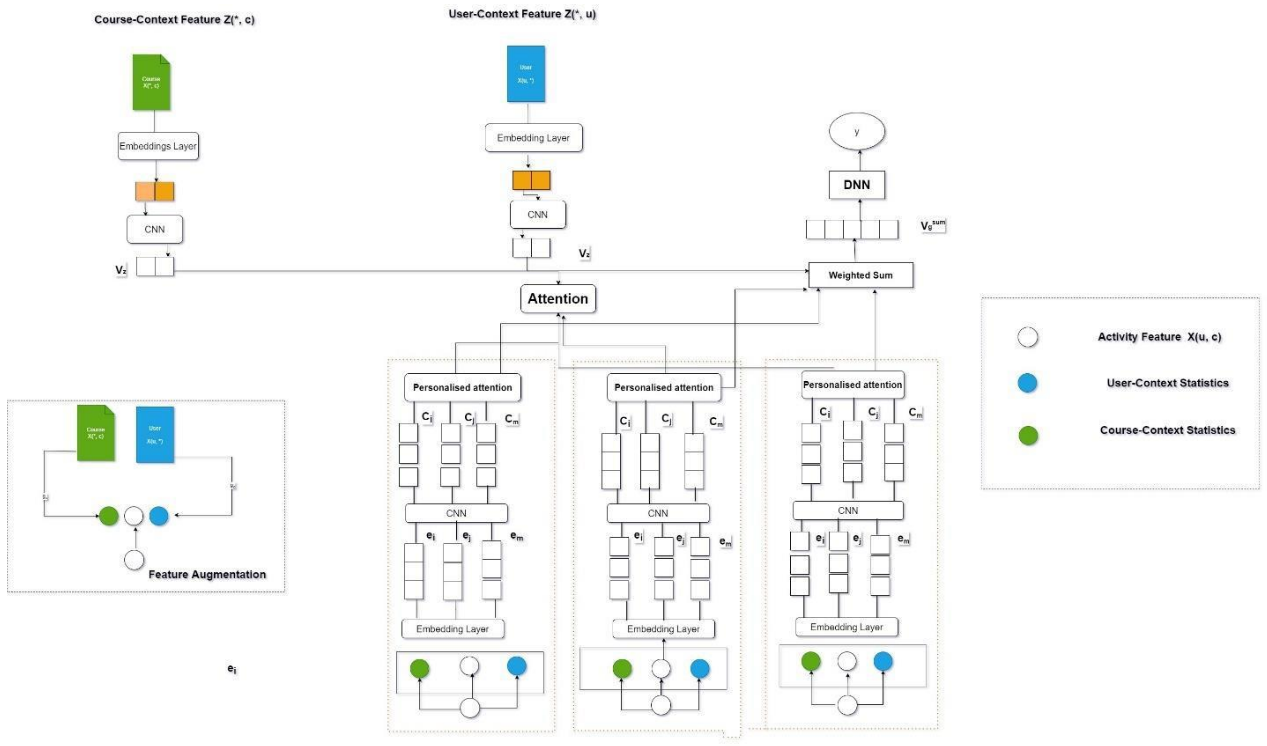
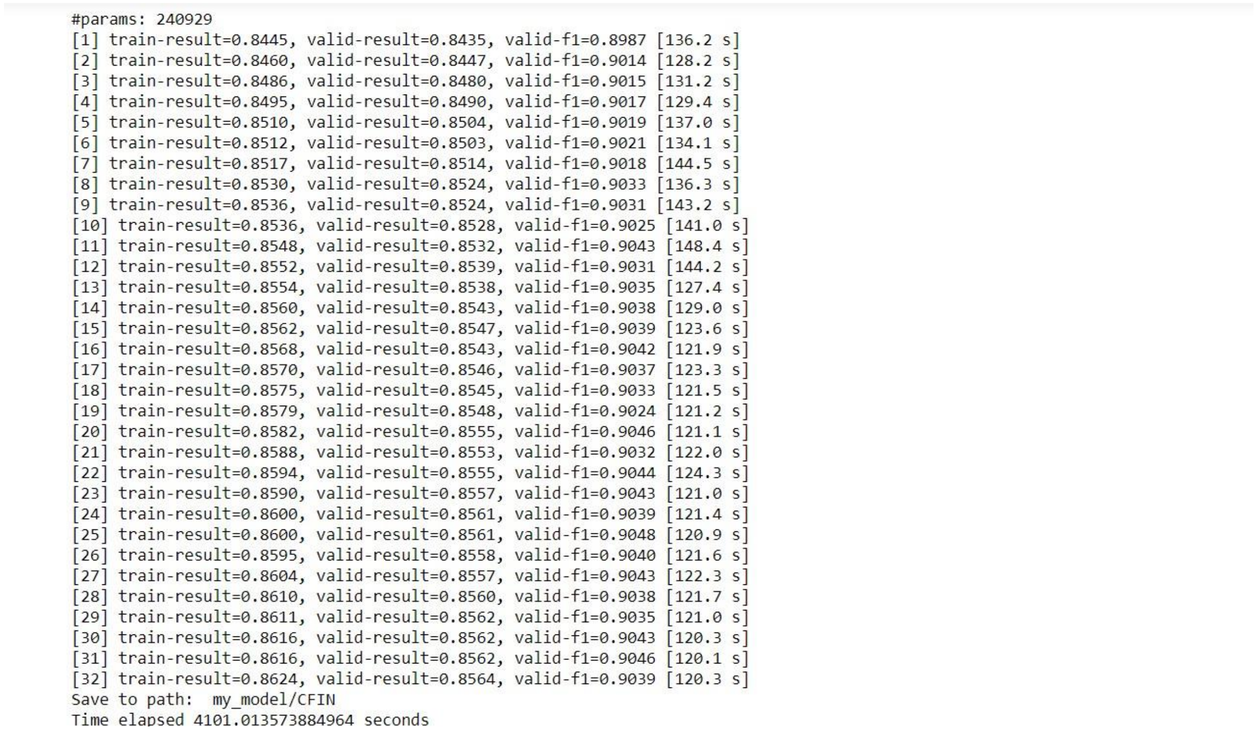


Figure 1: The Framework of CFIN Model for our Students Dropout Prediction Problem

3.8 Experiments

3.8.1 Dataset and Experimental Settings:

After doing systematical study of CFIN model approach used by (Feng, 2019), we have proposed some additional changes in the model. Firstly, to separate the user and course context from the model instead of (Feng, 2019) approach where they have embedded both user and course context. With this algorithmic change, we have observed that the model accuracy results has been achieved efficient comparatively with (Feng, 2019) such as during model training it gives F1-score as 90% during second epoch result.



```
#params: 240929
[1] train-result=0.8445, valid-result=0.8435, valid-f1=0.8987 [136.2 s]
[2] train-result=0.8460, valid-result=0.8447, valid-f1=0.9014 [128.2 s]
[3] train-result=0.8486, valid-result=0.8480, valid-f1=0.9015 [131.2 s]
[4] train-result=0.8495, valid-result=0.8490, valid-f1=0.9017 [129.4 s]
[5] train-result=0.8510, valid-result=0.8504, valid-f1=0.9019 [137.0 s]
[6] train-result=0.8512, valid-result=0.8503, valid-f1=0.9021 [134.1 s]
[7] train-result=0.8517, valid-result=0.8514, valid-f1=0.9018 [144.5 s]
[8] train-result=0.8530, valid-result=0.8524, valid-f1=0.9033 [136.3 s]
[9] train-result=0.8536, valid-result=0.8524, valid-f1=0.9031 [143.2 s]
[10] train-result=0.8536, valid-result=0.8528, valid-f1=0.9025 [141.0 s]
[11] train-result=0.8548, valid-result=0.8532, valid-f1=0.9043 [148.4 s]
[12] train-result=0.8552, valid-result=0.8539, valid-f1=0.9031 [144.2 s]
[13] train-result=0.8554, valid-result=0.8538, valid-f1=0.9035 [127.4 s]
[14] train-result=0.8560, valid-result=0.8543, valid-f1=0.9038 [129.0 s]
[15] train-result=0.8562, valid-result=0.8547, valid-f1=0.9039 [123.6 s]
[16] train-result=0.8568, valid-result=0.8543, valid-f1=0.9042 [121.9 s]
[17] train-result=0.8570, valid-result=0.8546, valid-f1=0.9037 [123.3 s]
[18] train-result=0.8575, valid-result=0.8545, valid-f1=0.9033 [121.5 s]
[19] train-result=0.8579, valid-result=0.8548, valid-f1=0.9024 [121.2 s]
[20] train-result=0.8582, valid-result=0.8555, valid-f1=0.9046 [121.1 s]
[21] train-result=0.8588, valid-result=0.8553, valid-f1=0.9032 [122.0 s]
[22] train-result=0.8594, valid-result=0.8555, valid-f1=0.9044 [124.3 s]
[23] train-result=0.8590, valid-result=0.8557, valid-f1=0.9043 [121.0 s]
[24] train-result=0.8600, valid-result=0.8561, valid-f1=0.9039 [121.4 s]
[25] train-result=0.8600, valid-result=0.8561, valid-f1=0.9048 [120.9 s]
[26] train-result=0.8595, valid-result=0.8558, valid-f1=0.9040 [121.6 s]
[27] train-result=0.8604, valid-result=0.8557, valid-f1=0.9043 [122.3 s]
[28] train-result=0.8610, valid-result=0.8560, valid-f1=0.9038 [121.7 s]
[29] train-result=0.8611, valid-result=0.8562, valid-f1=0.9035 [121.0 s]
[30] train-result=0.8616, valid-result=0.8562, valid-f1=0.9043 [120.3 s]
[31] train-result=0.8616, valid-result=0.8562, valid-f1=0.9046 [120.1 s]
[32] train-result=0.8624, valid-result=0.8564, valid-f1=0.9039 [120.3 s]
Save to path:  my_model/CFIN
Time elapsed 4101.013573884964 seconds
```

Figure 10CFIN Modelling Results

Secondly, we proposed how changes in user context features can impact the model accuracy results. We added user education level feature for context modelling and it increased the model performance. The below figure shows how user education level impacts their video activity in a course. After employeing the dataset from XuetanX MOOC platform, it shows that user's with bachelor's and Master's degree have the highest video watch time comparatively user's with associate level degrees. The results even found out the hidden age feature in the evaluation results. Thus, user's with bachelor's and Master's degree and who are in their early twenties age have the highest video activity time and lowest dropout rates in the MOOCS.

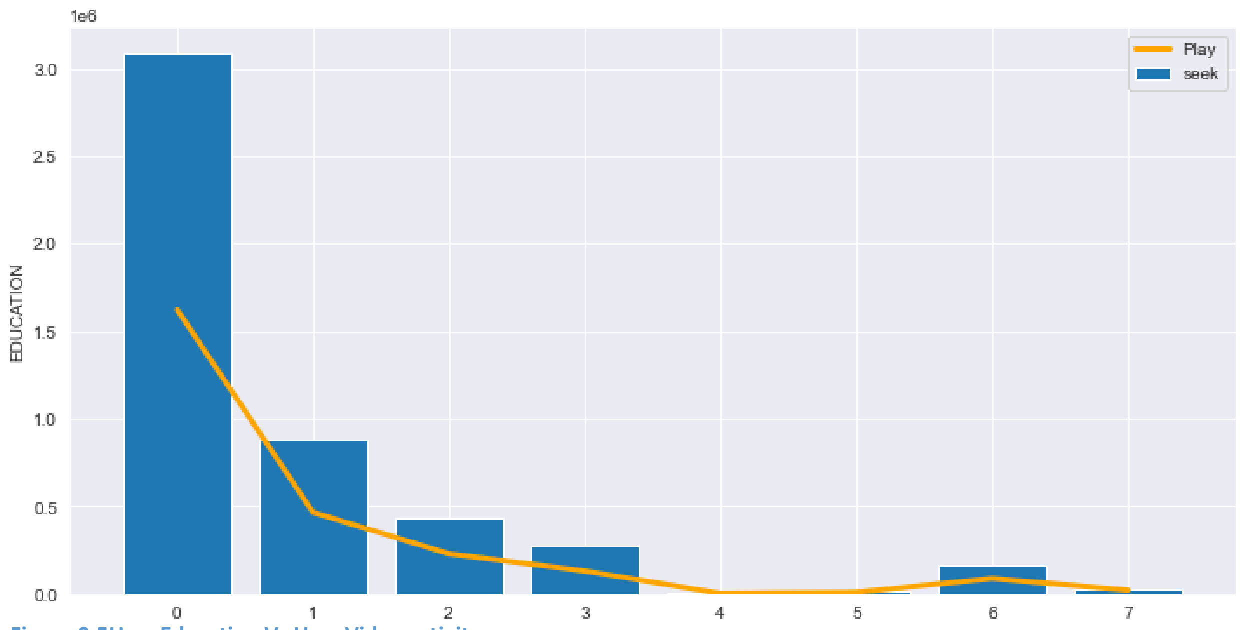


Figure 0:5User Education Vs User Video activity

Model Parameter Settings:

We used Tensorflow for the implementation of the CFIN model, and used attention mechanism similarly like CFIN model by (Feng, 2019) except using the embedded user and course context smoothing technique. We have chosen the adam optimized as it proved to be more effective for binaryclassification problems. Furthermore, while using embedding layering we have used the L2- regularziation technique to avoid overfitting and data sparsity problems. In this experiment, the embedding size has been set to 32, the convolution size as 512, DNN layers to 256, dropout rate has been chosen as 0.9, total batch size is set to 32, model learning rate is 0.0001. In the proposed model, we have used the dropout strategy to avoid overfitting in our results.

Parameter Settings
Embedding Size = 32
Attention Size = 16
Convolution Size = 512
Context Size = 32
DNN layers = 256
Dropout Rate = 0.9
Dropout Attention = 1.
Relu Activation Function = tf.nn.relu
Learning Rate = 0.0001
Optimizer = Adam
L2-regularization = 0.0001
Attention = True
Evaluation Metric = roc_auc_curve and F1-score
Random Seed = 12345
Verbose = True
Epochs = 32
Batch Normalization = 1

The adam optimizer has been used as a gradient decend function which has proven efficient for binary classification problems. The batch size is set to 32. The hyperparameters for the CFIN model are selected according to the validation set. For model evaluation, we have used AUC and F1-Score metrics.

3.9 Evaluation Metrics:

3.9.1 The ROC Curve:

The ROC curve score for the CFIN model is 0.8663. Which is still equivalent to CFIN model proposed by (Feng, 2019). But with this proposed approach, time elapse time is less with higher model output efficiency.

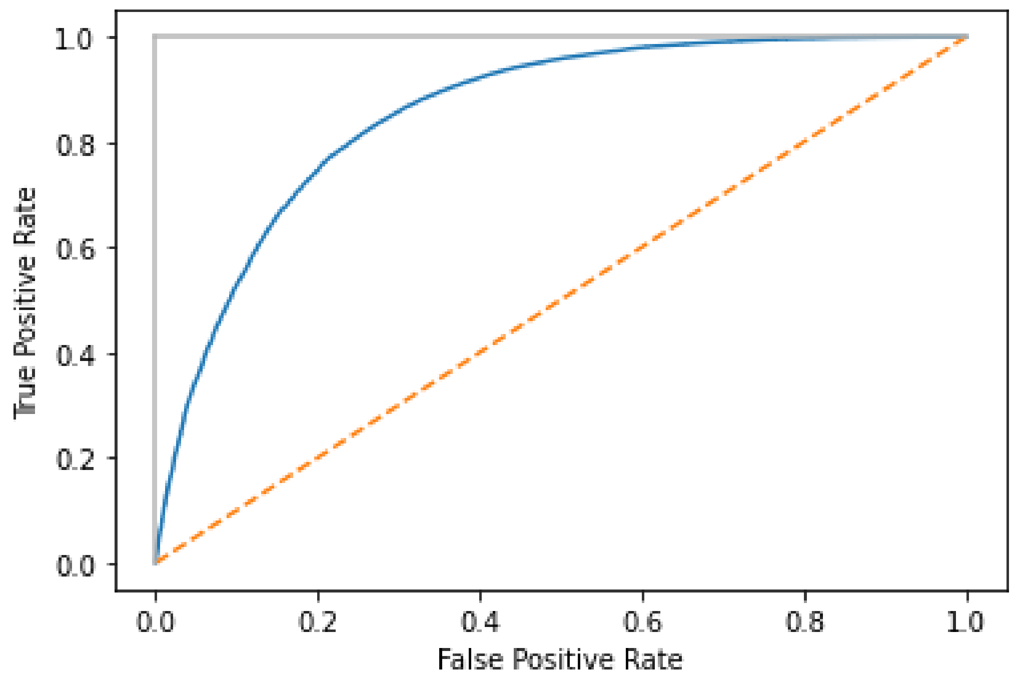


Figure 11CFIN Model: ROC Curve

3.9.2 F1-Score:

The F1-score is used to combine and measure the precision and recall in a single metric. The F1-score is known as the harmonic mean of recall and precision. Thus, harmonic mean gives more weight to low-values.

$$F_1 = \frac{2}{} = = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} \times \text{recall}} = \frac{\text{TP}}{\text{TP} + \frac{\text{FN} + \text{FP}}{2}}$$

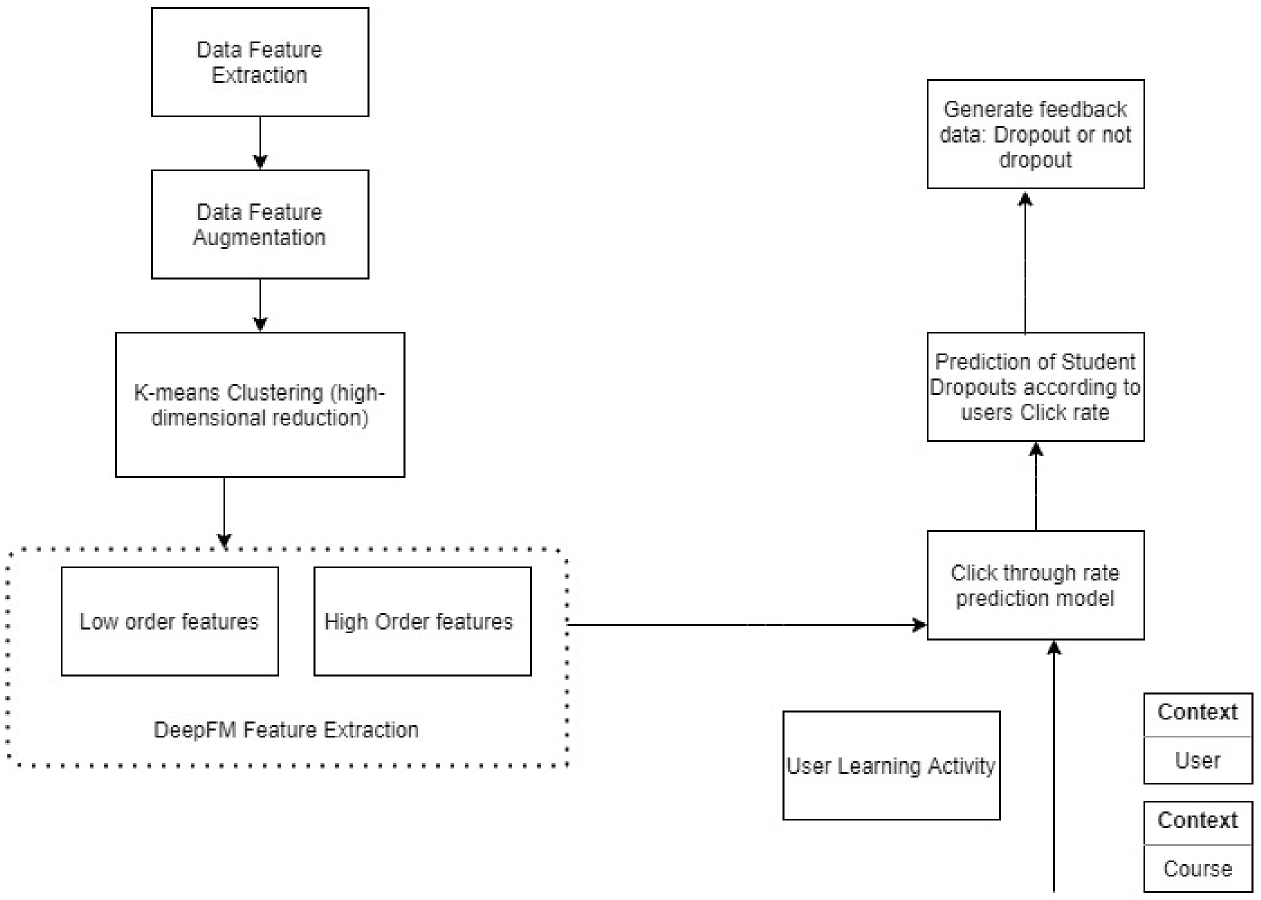
In this classifier, we have trained our model to predict students who are at higher risk of dropout and students with lowest rates of being dropout from a course. The proposed model achieves F1-score equals to 92 percent. (Note: in the code example sample dataset has been used which is why the performance results can vary a bit)

AUC Score	**F1- Score**
.8663	.9239

Chapter 4: The DeepFM model for CTR based Dropout Prediction in MOOCs

4.1 Introduction to DeepFM:

The prediction of student dropout rate through their clickstream data is critical for MOOC (Massive Open Online Courses) platforms to increase user retention, where the main goal is how to keep students engaged who are at the risk of dropouts. It is important for clickstream prediction to learn implicit features behind user clicks. These clickstream features are computed from the user’s learning activities that have all interaction events generated from the user behavior. The performance of a student dropouts prediction system depends on the mining of user click behavior relationships. Behavior relationships with personalized information help to better reveal the useful hidden information behind user click behavior, thus it is important for CTR (Click Through Rate) prediction to learn implicit feature interactions behind user click behaviors. This study aims at user’s learning activity for student's dropouts in XuetangX MOOC platform and proposes a personalized student retention method using a deep factorization machine (DeepFM) to learn implicit interactions behind user click behaviors on MOOCs dataset. The proposed method, DeepFM, integrates the power of factorization machines (FM) and deep neural network (DNN) for low-to-high level feature learning into a new neural network architecture.
With DeepFM parameter sharing strategy, the relationship between low order and high order feature interactions are learned from log data and can be trained end-to-end efficiently without any feature engineering and click rate prediction model is constructed. Finally, based on the predicted CTR, the area under the curve (AUC) and Logloss of the DeepFM method are 0.8323 and 0.3915 on XuetangX dataset.



Data Feature Extraction
Data Feature Augmentation
K-means Clustering (high-dimensional reduction)
Low order features
High Order features
DeepFM Feature Extraction
Generate feedback data: Dropout or not dropout
Prediction of Student Dropouts according to users Click rate
Click through rate prediction model
User Learning Activity
Context
User
Context
Course

Overall Framework of the DeepFM Method

4.2 Problem description and Algorithm framework:

Problem Description:

Understanding the student's dropout prediction in MOOCs can be regarded as a click-through rate prediction problem, in which the relationship between user features and user learning activity feature combinations plays a key role in student dropouts rate prediction. DeepFM has been widely used in CTR problems such as recommender systems, advertisement applications, agriculture, medical and so on. However, it has not yet been applied for student's dropouts in MOOCs problem. For instance, in recommender systems items are ranked based on the maximum number of CTR while for online advertising our goal is to maximize the revenue as well as estimation of correct CTR. Thus, correct estimation of CTR counting in MOOCs platforms does also provides an edge to improve student retention for MOOC platforms. Our main concern for the usage of DeepFM over other existing CTR models such as Factorization-Supported Neural Network (FNN), Product-based Neural Network (PNN), Wide and Deep is because some only capture low-order feature interactions while others capture high-order feature interactions and for some models like Wide and Deep will need extra feature engineering. Therefore, in the case of Student Dropouts problem we have examined user's learning activity for deep click-through rate prediction model based on feature combination relationships.

Algorithm Framework:
The proposed method, DeepFM, has two main stages: model training stage and prediction stage. For training the DeepFM model, we need to process the tracking log files that include all user's learning activities in the XuetangX platform to obtain a click-rate prediction model. Since the data may contain different discrete or continuous features, it is necessary to perform data augmentation to avoid overfitting or underfitting for correct CTR prediction. In the prediction stage, a trained CTR prediction model predicts the probability of a student dropout in a specific user context(age, gender, location, education) and course category context. Henceforth, correct estimation of CTR is a key to determine user dropout from a course. In Figure, we have displayed embedded user learning activity with its user and course context. Some of the dataset features will be used as sparse features such as Gender, education, cluster_label, course_category and others will be used as Dense Features.

The overall framework for the DeepFM model has been depicted in the diagram. Firstly, we will perform data extraction and data augmentation techniques to avoid overfitting in model.

Data Augmentation and Context Smoothing Strategy:
Data augmentation is necessary to enhance the neural network model performance on sparse and high data dimensionality. For effective deep learning models high-quality and abundant data is a key. For context smoothing, we will perform feature augmentation, embedding and feature fusion.

For user context: (Huifeng Guo, 2017)

$$\mathrm{g}_u(\mathrm{a}_i) : \mathrm{g}_u = \mathrm{A}_i(\mathrm{u},\mathrm{c}) \rightarrow [avg(\{\mathrm{A}_i(\mathrm{u},*)\}), \max(\{\mathrm{A}_i(\mathrm{u},*)\})]$$

For course context: (Huifeng Guo, 2017)

$$\mathrm{g}_c(\mathrm{a}_i): \mathrm{g}_c = \mathrm{A}_i(\mathrm{u},\mathrm{c}) \rightarrow [avg(\{\mathrm{A}_i(*,\mathrm{c})\}), \max(\{\mathrm{A}_i(*,\mathrm{c})\})]$$

For augmented feature activity: (J Xu, 2021)

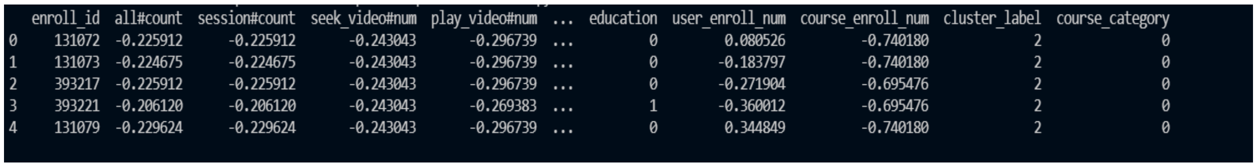
$$\hat{A} = \widehat{A_g^1} + \widehat{A_g^2} + \widehat{A_g^{m_x}}$$

Here each $A_i \in R^{m}{}_{g}$ is a feature group which will consist of feature activity A_i and its context statistics: (J Xu, 2021)

$$\widehat{A_g^i} = [[a_i] \oplus g_u(a_i) \oplus g_c(a_i)]$$

Definition: User Learning Activity:
In MOOCs, users learning activity in a course will be formulated as an m_x-dimensional vector $A(u, c)$ where u will represent user and c represents a course, where each element in $a(u,c) \in A(u, c)$ is a continuous feature value associated to user's u learning activity in course c.

We have transformed gender, education, user cluseter label, and course category into sparse features by coding. The original features input are one-hot or multi-hot types but they are transformed



```
   enroll_id  all#count  session#count  seek_video#num  play_video#num  ...  education  user_enroll_num  course_enroll_num  cluster_label  course_category
0     131072  -0.225912      -0.225912       -0.243043       -0.296739  ...          0         0.080526          -0.740180              2                0
1     131073  -0.224675      -0.224675       -0.243043       -0.296739  ...          0        -0.183797          -0.740180              2                0
2     393217  -0.225912      -0.225912       -0.243043       -0.296739  ...          0        -0.271904          -0.695476              2                0
3     393221  -0.206120      -0.206120       -0.243043       -0.269383  ...          1        -0.360012          -0.695476              2                0
4     131079  -0.229624      -0.229624       -0.243043       -0.296739  ...          0         0.344849          -0.740180              2                0
```

Figure 1Augmented User Learning Activity

We have also used k-means algorithm to reduce data dimension so that we can avoid overfitting in our model results. Because to reduce sparsity and noise in the dataset we are required to obtain low-dimensional representation from the high-dimensional log data. Afterwards, DeepFM model is used to extract low-order and high-order features from the low-dimensional log data. In the DeepFM model, a new embedding layer is introduced into the deep neural network (DNN) which compresses the input vectors into low-dimensional vectors and further reduces the data sparsity and noise. Finally, those extracted features are then trained for CTR prediction model.

4.3 Proposed Personalized Product Recommendation Method:

4.3.1 DeepFM:

DeepFM is an end-to-end learning model, and it consists of two parts namely as Factorization Machines (FM) and Deep Neural Network (DNN), and they share the same input. In this model, we aim to learn both low-order and high-order feature interactions. For a certain feature i, a scalar w will represent the weight of i-th feature, and a vector V_i will be used to represent k-dimensional implicit vector of i-th feature. Here, V vector is used as an input into the FM and DNN part. In the FM part, latent vector V_i is combined with other features to measure its impact of interactions with other features. Basically, we are using vector V_i to be fed into FM component to fit second-order feature interactions. While vector V_i is fed into DNN component to model high-order feature interactions where high-order feature interaction relationship is learned through a neural network and nonlinear feature combinations. Parameters for the neural network and factorization machines will be trained jointly for the combined prediction model.

$$\hat{y} = sigmoid\ (y_{FM} + Y_{DNN})$$

Where $\hat{y} \in (0, 1)$is the click-rate prediction value whereas Y_{FM} is the output of Factorization machine and Y_{DNN} is the output of deep neural network (Huifeng Guo, 2017).

Factorization Machines (FM):

In the FM part, our main goal is to learn order-2 feature interaction relation and the first order weights. With decomposer technique, capturing order-2 feature interactions is more effective as compared to previous approaches. The major benefit of using FM technique is its flexibility of providing inner product of implicit V_i

and V_j feature vector. When the dataset is sparse, it can capture order-2 feature combinations much more effectively through inner product of latent vectors. Therefore, decomposer model is helpful to be trained and learnt in situations where data is sparse and feature interactions rarely appear in the training data.

$$Y_{FM}(x) = < w, x > + \sum_{j=i+1}^{d} < V_i , V_j > x_i . x_j$$

In above equation, <w, x> represents the addition unit and weights of the order-1 feature interactions. While $< V_i , V_j >$ represents the inner product unit and the weights of order-2 feature interactions. As shown in sigmoid equation, $Y_{FM}(x)$ factorization machines output is used as a part of CTR prediction.

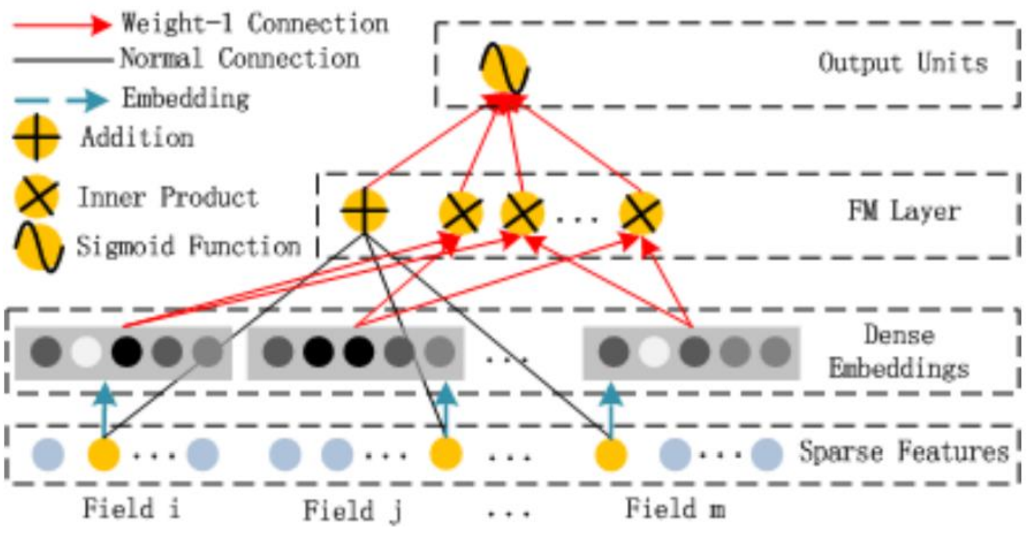


Figure 0-1 The FM Component Architecture (Huifeng Guo, 2017)

In the FM component, plus circle represents the weights of order-1 feature interactions. In the FM architecture figure, Sparse Features are connected by a black line and the weight is W_i. Here, cross circles represent the inner product $< V_i, V_j>$ and it shows its correspondence to interactive feature calculation between each feature. In the model dense embedding layer is introduced that is used to convert the discrete vector to low-dimensional dense vector. In the figure, dense embedding layer is connected with red line that will use the dictionary, feature index matrix, and feature value matrix to calculate feature combinations.

Deep Neural Network (DNN):

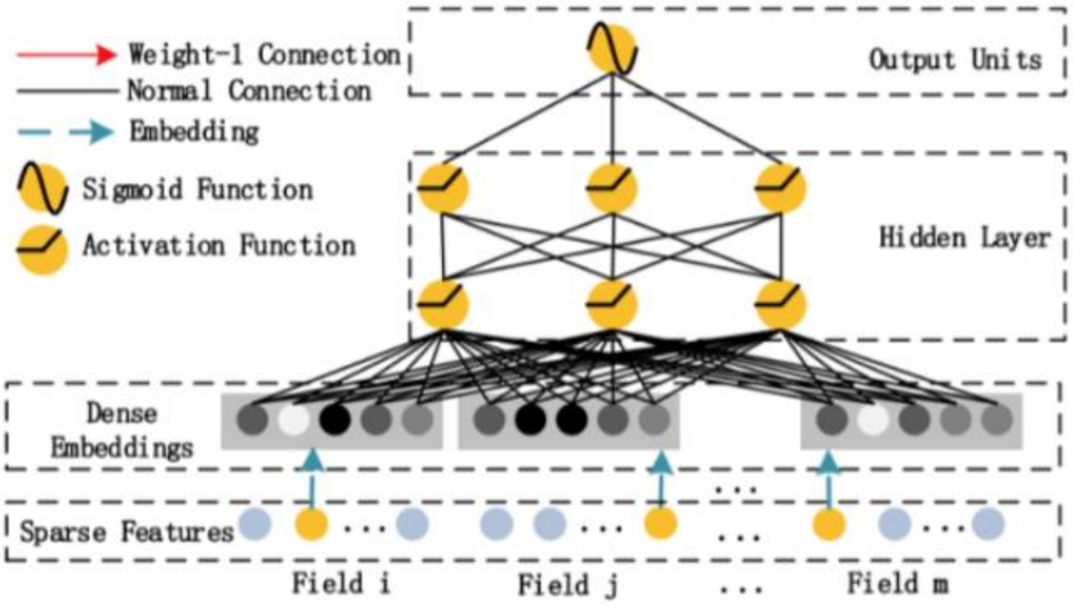


Figure 0-2The DNN component architecture (Huifeng Guo, 2017)

DNN is considered as a feed-forward neural architecture, which is basically used to learn high-order feature interactions. (Huifeng Guo, 2017). The original data for CTR prediction, user tracking log files in XuetangX platform, is highly sparse and high-dimensional and grouped in fields such as age, gender, location, education, course category and user cluster labels. This suggests that data with categorical and continuous mixed with high sparsity will result in difficulty for parameter training of DNN. Additionally, the model training result will result in overfitting. To solve high-sparsity problem, our proposed model uses embedding layer to compress the high-dimensional input vector into low-dimensional, dense real value vector. Afterwards, the model will connect dense embedding layer with hidden layer of neural network. (J Xu, 2021)

In the DNN architecture figure, embedding layer has been highlighted. There are two interesting features of embedding layer:

(1) Although, lengths of different domains can be different, but their embedding size will be same. (Huifeng Guo, 2017)

(2) The implicit vector in FM now serves as network weights. Further, these weights will be used to compress the input vector into embedding layer.

DeepFM eliminated the need to pre-train FM separately and instead it will be used just as a part of DeepFM and will be trained together in an end-to-end manner with other parameters. (J Xu, 2021)

We will denote the embedding layer as follows:

$$a^0 = [\, e_1 \,, e_2, e_3, \dots, e_m \,]$$

In the above embedding layer equation, e_i represents the embedding of i-th field, and m will represent the number of domains. After defining embedding layer, we will feed the embedding output to deep neural network.

$$a^{(l+1)} = \sigma(W^l a^l + b^l)$$

Where l : layer depth, σ : activation function , a^l: output of embedding layer, W^l: model weight, b^l : bias of the l-th layer.

After feeding embedding layer output to neural network, we will generate a dense real-value feature vector. At the final step, that dense vector will be fed into the sigmoid function for CTR prediction.

$$Y_{DNN} = W^{|H|} . a^{|H|} + b^{|H|+1} \;\; ; \;\; where\ |H| : no\ of\ hidden\ layers$$

In a nutshell, DeepFM model provide two main advantages. Firstly, embedding vector will learn both low and high-order feature interactions from raw dataset. Secondly, there is no need of feature engineering.

4.4 Experimental Results and Analysis:

4.4.1 Experimental Dataset:

XuetangX MOOC Platform Dataset:

To do experiments on Students Dropouts Prediction in MOOCs, we will use XuetangX dataset, one of the largest MOOCs provider in China. XuetangX has provided over 1000+ courses and enrolled 10,000,000 users. XuetangX is basically providing courses from different categories such as computer science, engineering, history, math, physics etc. Added to that, they have also provided two learning modes to take course, users can choose Instructor-based mode (IPM) or self-based mode (SPM). XuatangX provides richer data information such as it provides users tracking log files which contains user click-through rate data such as webpage clicking (click and close a course), video watching (play, stop, pause, jump), forum discussion (ask questions and replies) and assignment completion (correct, incorrect, or reset answers). DeepFM model is not yet applied to a MOOC students dropout problem. This will be the first experiment with MOOCs Dataset using DeepFM model.

Sparse Features: Gender, education, cluster_label, course_category

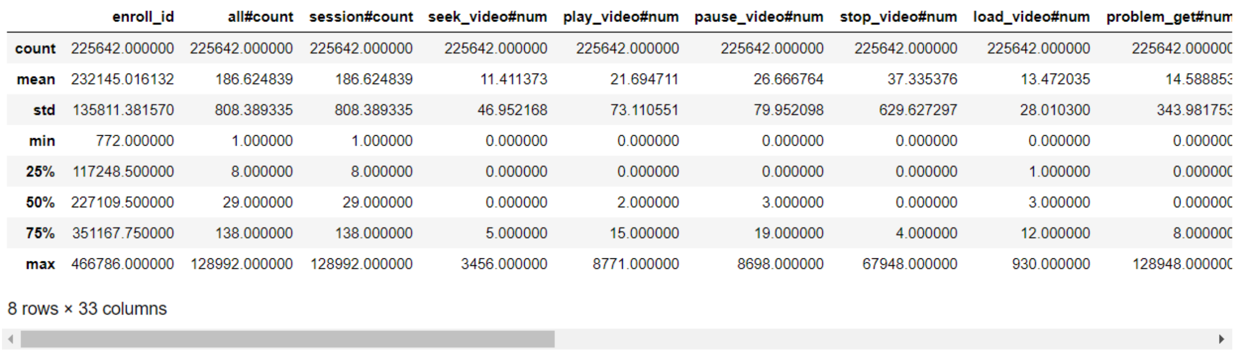
User Data	**Course Data**
Gender education User cluster label	Course category Course type

Dense Features: User learning activity features will be used as dense features in DeepFM Model

Video	Assignment	Forum	Click Action	Enrollment
#seek_video #play_video #pause_video #stop_video #load_video	#problem_get #problem_check #problem_save #reset_problem #problem_check_correct #problem_check_incorrect	#create_thread #create_comment #delete_thread #delete_comment	#click_info #click_courseware #click_about #click_forum #click_progress	#enrollment total#users dropout_rate

The XuetangX dataset contains richer information, and it has 698 IPM courses and 515 SPM courses. Also, the dataset is also very huge to assess model accuracy. The dataset has 5 discrete features such as gender, education, user cluster label, course type, course category and 24 continuous features. Here, discrete data will be used to meet the input requirements of the model while continuous data will need to be normalized to increase the effectiveness of model learning and to avoid model overfitting. For data sparsity issues, index matrices and dictionaries will be used for discrete and continuous data types and merged.

The dataset is the user click log data embedded with user and course context. The training dataset has 80 percent of full data while the testing dataset will have 20 percent of dataset. The test dataset is consistent with training dataset. Furthermore, not to affect the studies null values have been cleaned and removed from the dataset. Descriptive statistical analysis has been displayed in the below image for MOOC Dataset.



	enroll_id	all#count	session#count	seek_video#num	play_video#num	pause_video#num	stop_video#num	load_video#num	problem_get#num
count	225642.000000	225642.000000	225642.000000	225642.000000	225642.000000	225642.000000	225642.000000	225642.000000	225642.00000
mean	232145.016132	186.624839	186.624839	11.411373	21.694711	26.666764	37.335376	13.472035	14.58885
std	135811.381570	808.389335	808.389335	46.952168	73.110551	79.952098	629.627297	28.010300	343.98175
min	772.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	117248.500000	8.000000	8.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.00000
50%	227109.500000	29.000000	29.000000	0.000000	2.000000	3.000000	0.000000	3.000000	0.00000
75%	351167.750000	138.000000	138.000000	5.000000	15.000000	19.000000	4.000000	12.000000	8.00000
max	466786.000000	128992.000000	128992.000000	3456.000000	8771.000000	8698.000000	67948.000000	930.000000	128948.00000

8 rows × 33 columns

Figure 0-1 Descriptive Statistics for User Learning Activity Dataset

4.5 Evaluation Indicators:

For evaluation purpose, we will be using two evaluation indexes, namely Area Under the Curve (AUC) and logarithmic loss function (logloss).

4.5.1 The ROC Curve:

The area under the curve or receiver operating characteristics (ROC) is used for binary classification problems. The ROC curve plots the true positive rates against the false positive rates. The ratio of negative instances that are incorrectly classified as positive are known as false positive rate. On other hand, the ratio of negative instances correctly classified as negative will be considered as true positive rate. True positive rate is also known as recall, and the roc curve plots the true positive rate(recall) versus false positive rate. In ROC, the higher the true positive rate will produce the more false positive rates.

ROC Equation: (Huifeng Guo, 2017)

$$\boldsymbol{AUC} = \frac{\sum_{i\,\in positive-class} \boldsymbol{rank_i} - {\boldsymbol{M} \times (\boldsymbol{M}+1)}/{2}}{\boldsymbol{M} \times \boldsymbol{N}}$$

Now, we will explain the Area Under the Curve equation as follows:

M : Positive sample number
N : Negative sample number
rank$_i$: Sample i in a given model in ascending order

Here, AUC will be defined as a positive and negative sample are selected at random and we will further use a classifier to rank the positive sample in front of negative sample.

4.5.2 LogLoss Function:

The logarithmic loss function is used for binary classification models where it measures the performance of the model whose output value is either 1 or 0. With logloss cost function, the model estimates the high probabilities for positive samples (y = 1) and low probabilities for negative samples (y = 0). If the predicted probability diverges from the actual label, the loss will increase. For instance, when actual observation label is 1 and the predicted probability is 0.012 will result in high-loss value. However, a perfect model will have cross entropy loss of 0.

Math: (O'Reilly, 2019)

$$\boldsymbol{C}(\boldsymbol{\theta}) = \begin{cases} -log\,(\widehat{\boldsymbol{p}}) & if\ \ y=1 \\ -log(1-\widehat{\boldsymbol{p}}) & if\ y=0 \end{cases}$$

In the equation, -log(p) will grow very large as p is very close to 0 for positive instance or 1 for negative instance. So, when p is close to 1 our -log(p) will be close to 0. Hence, cost will be close to 1 for positive instance and close to 0 for negative instance. (O'Reilly, 2019)

$$e(x,y) = -\left(y.\log\left(\hat{y}(x)\right) + (1-y).\left(1-\log\left(1-\hat{y}(x)\right)\right)\right)$$

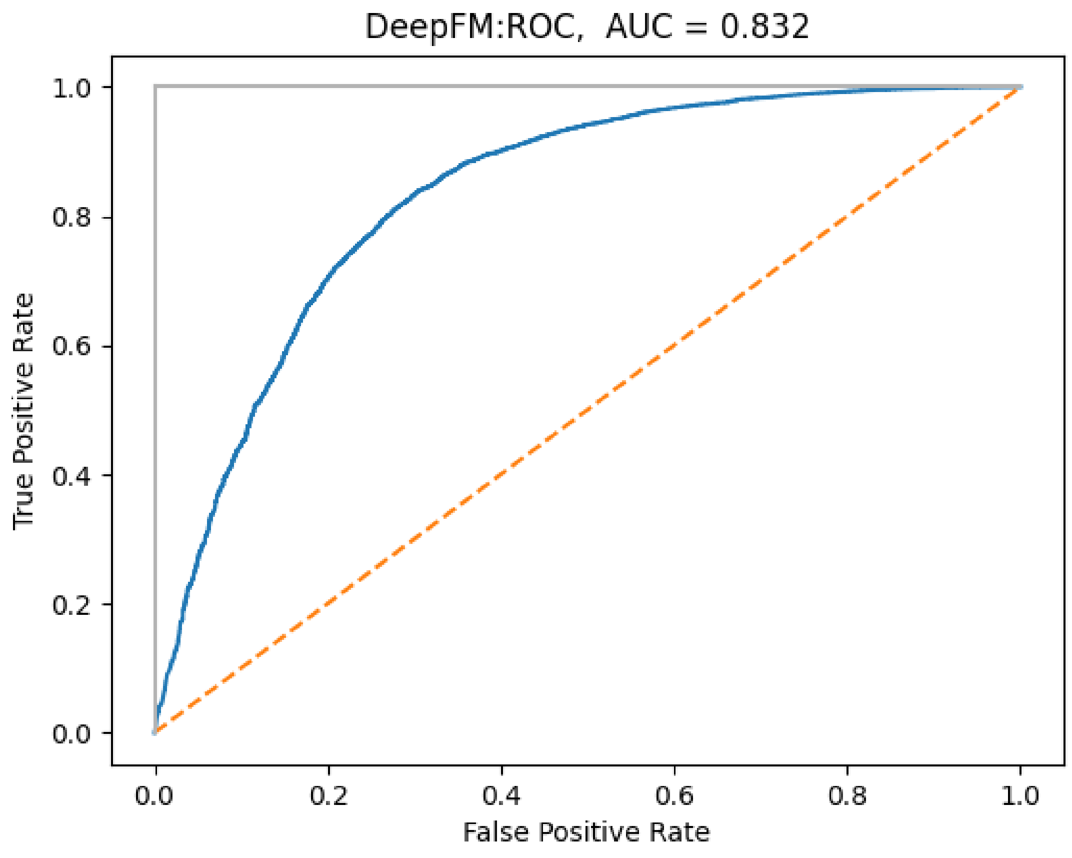
$$E = \sum(x,y)\epsilon\, T\; e(x,y)$$

(x, y) : Data sample where x is feature vector and y is label
Y(x): Predicted value of the input vector
e(x, y): logloss cost function for the predicted and correct result
E: logloss function of the entire dataset T

Implementation Details:
To plot ROC curve, use roc_curve() method to compute true positive rate and false positive rates.

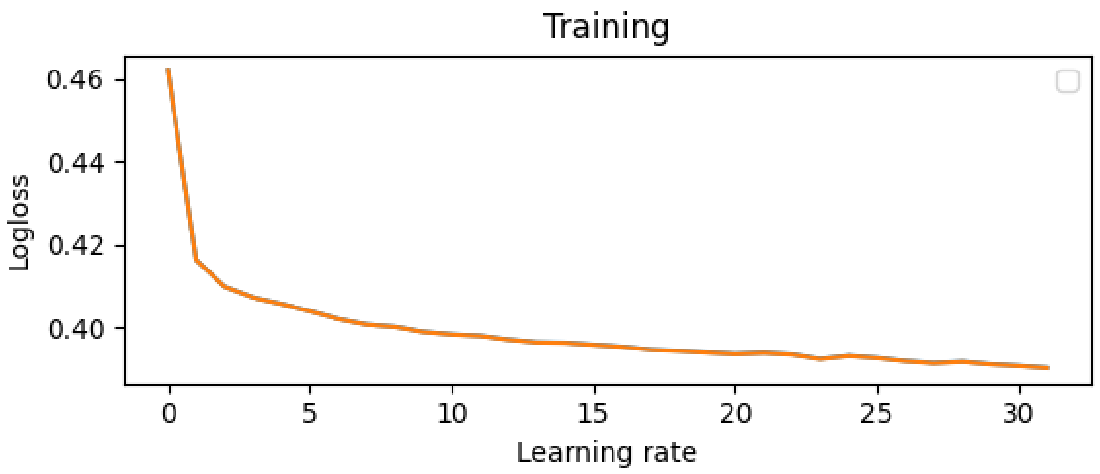
```
fpr, tpr , threshold = roc_curve(test[target].values, score1)
pyplot.title('DeepFM:ROC,  AUC = 0.832')
pyplot.plot(fpr, tpr)
pyplot.plot([0, 1], ls="--")
pyplot.plot([0, 0], [1, 0] , c=".7"), pyplot.plot([1, 1] , c=".7")
pyplot.ylabel('True Positive Rate')
pyplot.xlabel('False Positive Rate')
pyplot.show()
```

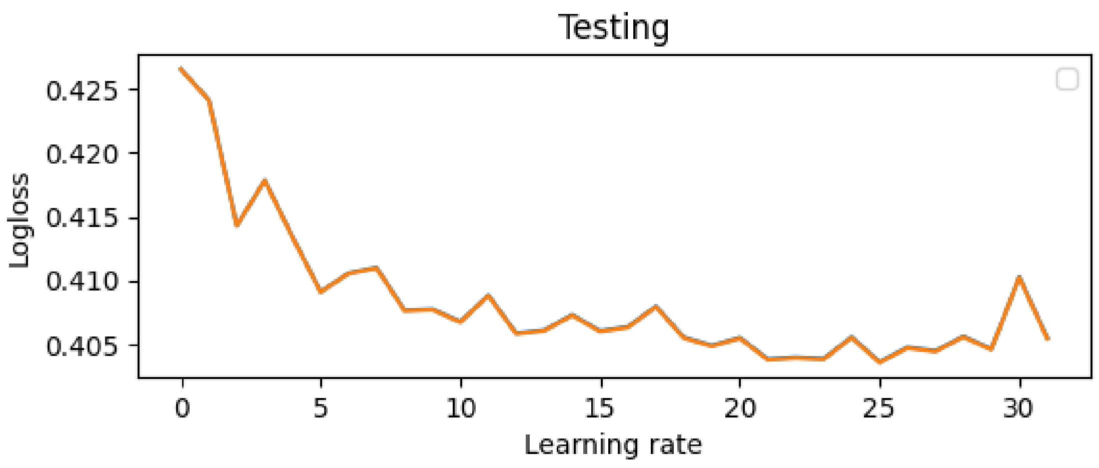
In this ROC curve, it plots the false positive rates against the true positive rates for all possible thresholds. We have also computed the AUC score which is AUC = 0.832 and its logloss = 0.3919.



AUC = 0.832	**Logloss = 0.3919**

Here, is the practical implementation of logloss for user learning activity embedded with user and course context using DeepFM method. The logloss score for training dataset is around 0.46 maximum while for testing dataset the logloss score lies around 0.42.

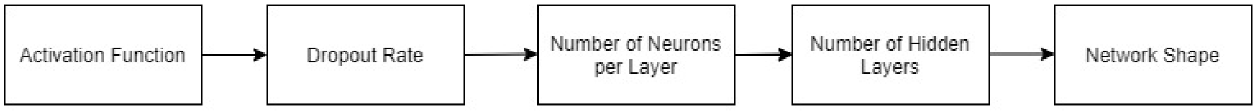




Parameter Performance Analysis:
Before this research study, DeepFM model has not been applied for MOOCs student dropout prediction. In this research study, we will use the available DeepCTR model package to get results for DeepFM model for student dropouts prediction based on click through rate. This model provides a strong click through rate algorithm and easily scalable for any dataset by using the exisiting components. In this study, first goal is to set the hyper-parameters f of the proposed model, deepFM, with its learning curve. Further,

HYPER PARAMETER LEARNING:

In this section, we will explore the different hyperparameter settings in deepFM model and their impact on model performance using XuetangX MOOCs dataset.



Hyperparemeter Tuning Order

Hyper Parameters	**Definitions**
linear_feature_columns	An array of features used by linear part of model
dnn_feature_columns	An array of features used by dense part of model
fm_group	List of feature that will be used for feature interactions in the model
dnn_hidden_units	List of positive integer for DNN’s layer
L2_reg_linear	Applied l2 regularization to the linear part
L2_reg_embedding	Applied l2 regularization to the embedding vector
L2_reg_dnn	Applied l2 regularization to the DNN
Seed	Random seed value
dnn_dropout	[0,1) dropout probability
Dnn_activation	Activation function such as Sigmoid, tanH etc
Dnn_use_bn	Batch normalization set as true or false
Task	Binary task for binary classiification problem

To avoid overfitting, l2 regularization of 0.00001 has been added to the order-1 and order-2 feature interactions. The embedding dimension has been set as 32. The learning rate has been set as 0.0001 and number of neurons in dense part has been set as 256. The dropout rate is 0.9 for the model and batch size is 256. In final step, Adam optimizer function has been chosen for model optimization.

Activation Function:

We have chosen sigmoid activation function for binary classification problem (0 or 1). Sigmoid function is especially useful for problems where we are required the probability as 0 or 1 output. Since dropout prediction is a binary classification approach, thus sigmoid is the right choice.

Learning Rate: If the learning rate is too small or too large it would not provide the optimal value. To accuratly estimate the optimal value choosing an appropriate learning rate would be very effective approach.

Learning rate = 0.0001	Logloss = 0.3953	AUC: Accuracy = 0.8322

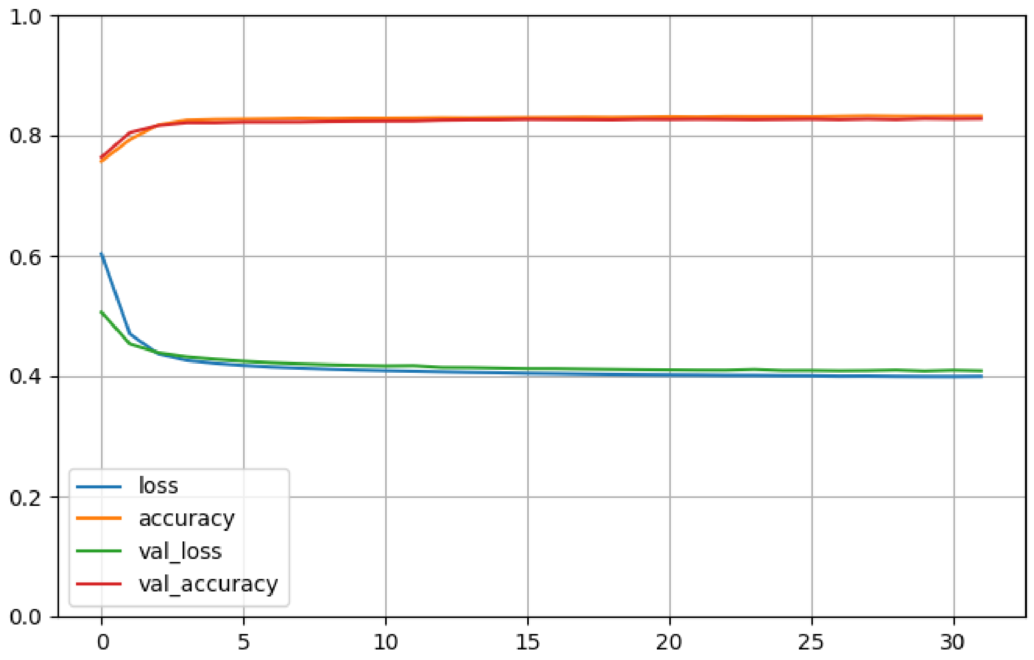


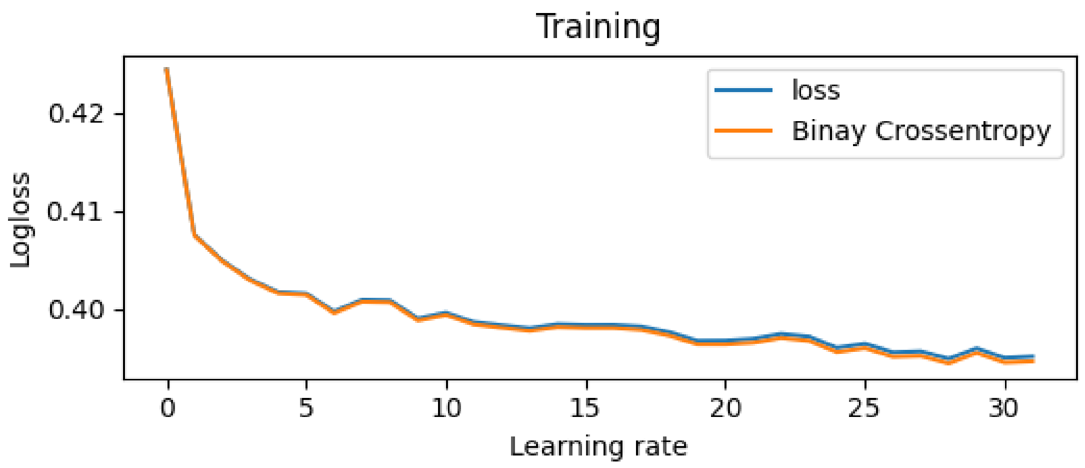
Figure 0-2Learning Curve: The mean training logloss and accuracy has been measured over each epoch and also the mean of validation logloss and validation accuracy is measured.

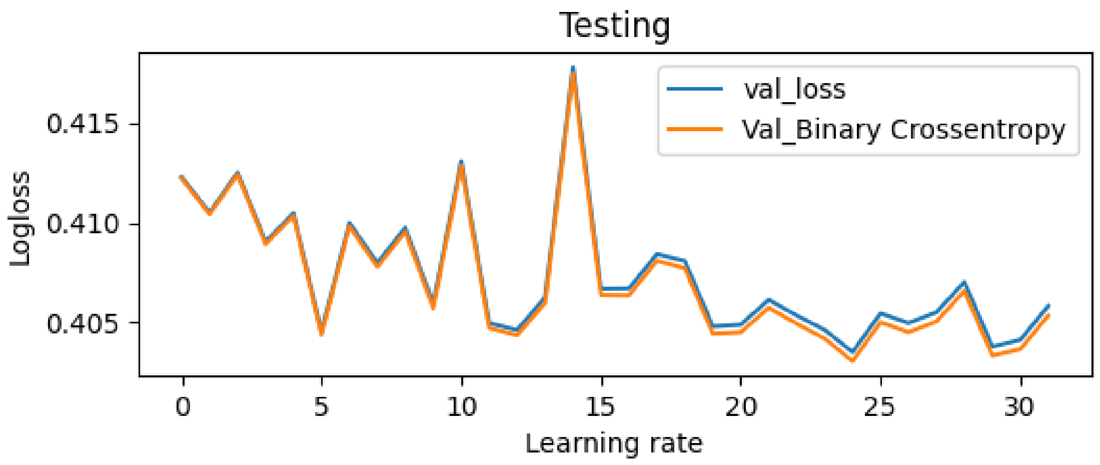
Learning rate = 0.001	Logloss = 0.392	AUC: Accuracy = 0.8411

By decreasing the learning rate value from 0.0001 to 0.001, we get reduced logloss function and increased accuracy value to 84 percent. While setting learning rate to 0.01 will fail to provide optimal output values.

Learning rate = 0.01	Logloss = 0.3918	AUC: Accuracy = 0.83

It has decreased the loss value, but it also reduced model accuracy score from 84 to 83 percent. By adjusting the parameter weights and values, the model AUC score improves to 83% with logloss as 0.3911. Albeit, the logloss value is still very same with default parameter values but these changes are still improving the model accuracy in a smaller range.


Training
loss
Binay Crossentropy
Logloss
0.42
0.41
0.40
0
5
10
15
20
25
30
Learning rate


Testing
val_loss
Val_Binary Crossentropy
Logloss
0.415
0.410
0.405
0
5
10
15
20
25
30
Learning rate

Chapter 5: Ensemble Learning for Dropout Prediction Problem

Different Deep Learning Models for Students Dropout Prediction in MOOCS:

Researchers have proposed various prediction methods in log files. So far, machine learning has been proven very effective to handle predictions based on user learning activity log files. However, due to data sparsity and overload information it has been noticed that there are some reliability and baseline model overfitting issues. (E. Mulyani, 2019) In this section, we will use three classifiers such as Support Vector Machine, Random Forest, and Decision Tree Model for MOOCS dropout prediction problem. We achieve 83% accuracy results with decision tree classifier. Furthermore, we will dig down to ensemble modeling and smote analysis techniques and will do comparative study of these techniques over baseline models. After doing experimental study with baseline classifier, we analyzed that for click through rate log files using single classifier can encounter problems such as imbalance classes, model overfitting, feature extraction problems. For this problem, synthetic minority over sampling (SMOTE) and ensemble learning approaches can be used to improve performance prediction results. The figure below, provides the overall framework using ensemble modelling techniques.

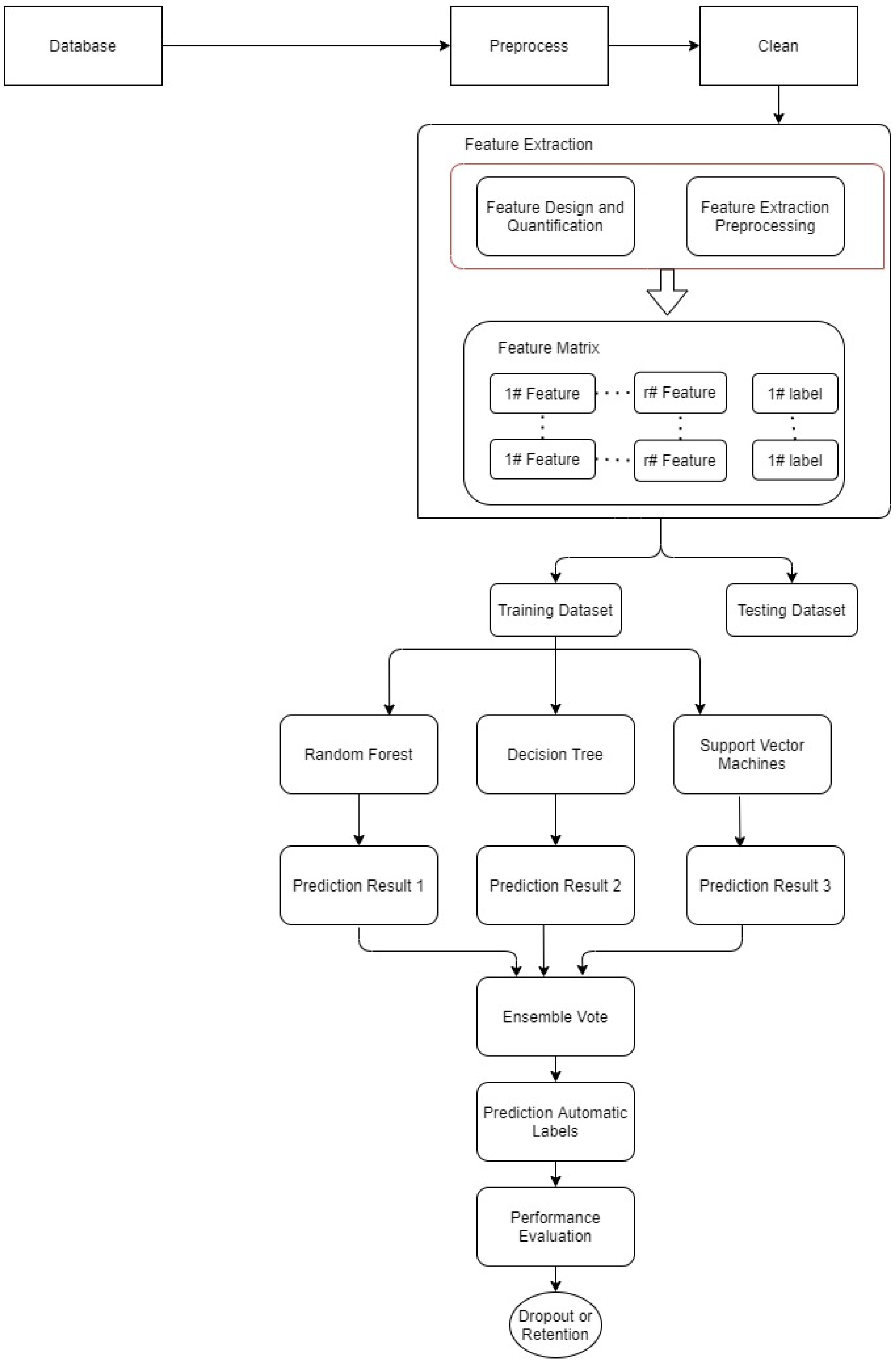
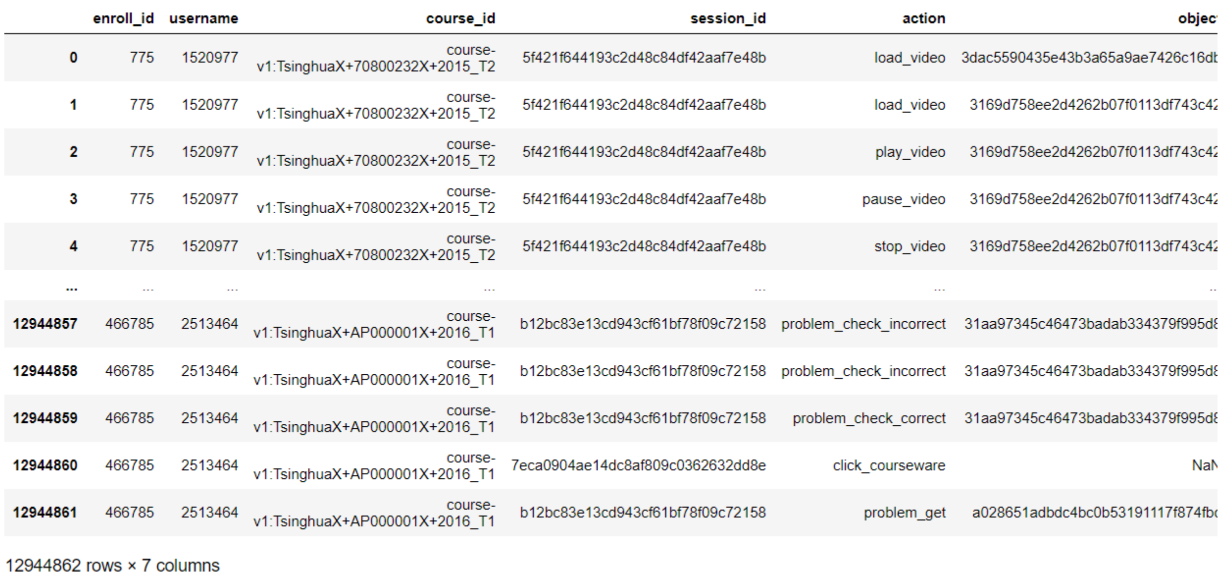


Figure 6OOverall Framework of Dropout Prediction Problem

5.1 Dataset Description:

Data preprocessing techniques are same as applied in DeepFM and CFIN modelling approach. A complete data description has already been provided above. Additionally, for feature selection we have completed the comparison importance of different features with truth tables in the dataset.

To summarize the steps, we have displayed our raw log files dataset.

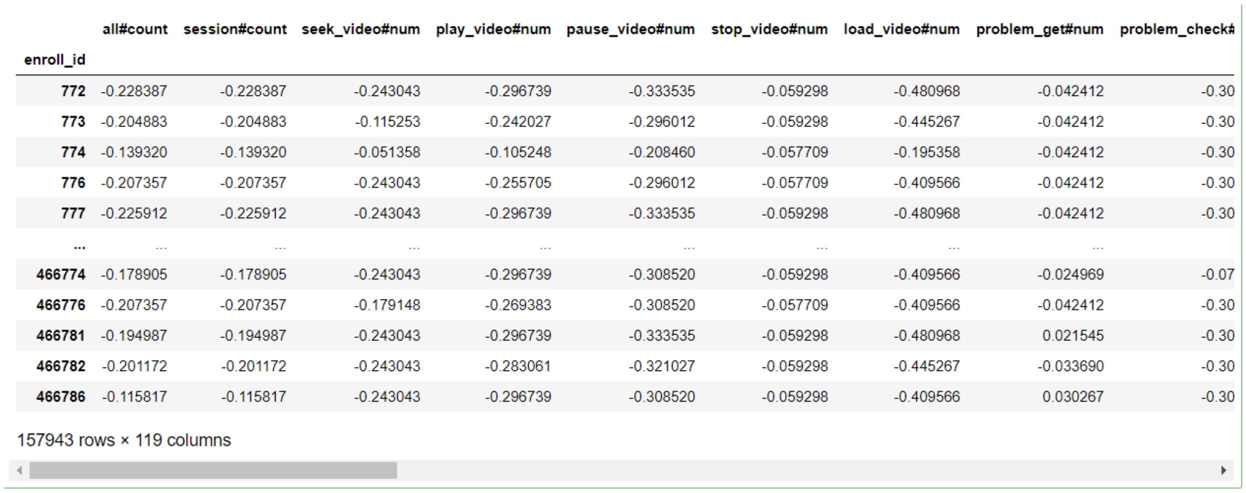


	enroll_id	username	course_id	session_id	action	objec
0	775	1520977	course-v1:TsinghuaX+70800232X+2015_T2	5f421f644193c2d48c84df42aaf7e48b	load_video	3dac5590435e43b3a65a9ae7426c16db
1	775	1520977	course-v1:TsinghuaX+70800232X+2015_T2	5f421f644193c2d48c84df42aaf7e48b	load_video	3169d758ee2d4262b07f0113df743c42
2	775	1520977	course-v1:TsinghuaX+70800232X+2015_T2	5f421f644193c2d48c84df42aaf7e48b	play_video	3169d758ee2d4262b07f0113df743c42
3	775	1520977	course-v1:TsinghuaX+70800232X+2015_T2	5f421f644193c2d48c84df42aaf7e48b	pause_video	3169d758ee2d4262b07f0113df743c42
4	775	1520977	course-v1:TsinghuaX+70800232X+2015_T2	5f421f644193c2d48c84df42aaf7e48b	stop_video	3169d758ee2d4262b07f0113df743c42
...	...	...	...	...	...	..
12944857	466785	2513464	course-v1:TsinghuaX+AP000001X+2016_T1	b12bc83e13cd943cf61bf78f09c72158	problem_check_incorrect	31aa97345c46473badab334379f995d8
12944858	466785	2513464	course-v1:TsinghuaX+AP000001X+2016_T1	b12bc83e13cd943cf61bf78f09c72158	problem_check_incorrect	31aa97345c46473badab334379f995d8
12944859	466785	2513464	course-v1:TsinghuaX+AP000001X+2016_T1	b12bc83e13cd943cf61bf78f09c72158	problem_check_correct	31aa97345c46473badab334379f995d8
12944860	466785	2513464	course-v1:TsinghuaX+AP000001X+2016_T1	7eca0904ae14dc8af809c0362632dd8e	click_courseware	NaN
12944861	466785	2513464	course-v1:TsinghuaX+AP000001X+2016_T1	b12bc83e13cd943cf61bf78f09c72158	problem_get	a028651adbdc4bc0b53191117f874fbc

12944862 rows × 7 columns

Figure 7Users Tracking Log File

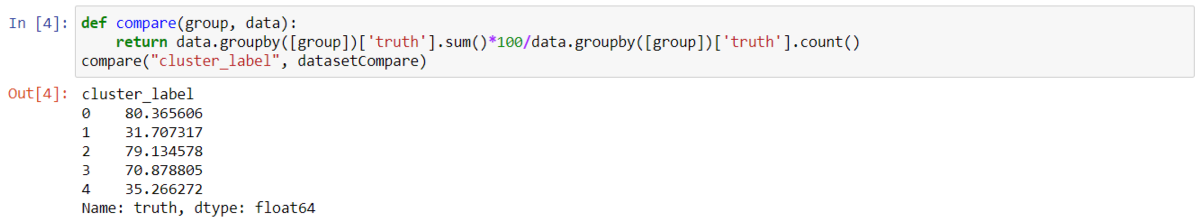
After preprocessing the dataset which contains four tables: User Information, Course Information, Truth Tables, and Users' Tracking log files. We will get our preprocessed dataset as shown below in the table.



enroll_id	all#count	session#count	seek_video#num	play_video#num	pause_video#num	stop_video#num	load_video#num	problem_get#num	problem_check#
772	-0.228387	-0.228387	-0.243043	-0.296739	-0.333535	-0.059298	-0.480968	-0.042412	-0.30
773	-0.204883	-0.204883	-0.115253	-0.242027	-0.296012	-0.059298	-0.445267	-0.042412	-0.30
774	-0.139320	-0.139320	-0.051358	-0.105248	-0.208460	-0.057709	-0.195358	-0.042412	-0.30
776	-0.207357	-0.207357	-0.243043	-0.255705	-0.296012	-0.057709	-0.409566	-0.042412	-0.30
777	-0.225912	-0.225912	-0.243043	-0.296739	-0.333535	-0.059298	-0.480968	-0.042412	-0.30
...	...	...	...	...	...	...	...	...	
466774	-0.178905	-0.178905	-0.243043	-0.296739	-0.308520	-0.059298	-0.409566	-0.024969	-0.07
466776	-0.207357	-0.207357	-0.179148	-0.269383	-0.308520	-0.057709	-0.409566	-0.042412	-0.30
466781	-0.194987	-0.194987	-0.243043	-0.296739	-0.333535	-0.059298	-0.480968	0.021545	-0.30
466782	-0.201172	-0.201172	-0.243043	-0.283061	-0.321027	-0.059298	-0.445267	-0.033690	-0.30
466786	-0.115817	-0.115817	-0.243043	-0.296739	-0.308520	-0.059298	-0.409566	0.030267	-0.30

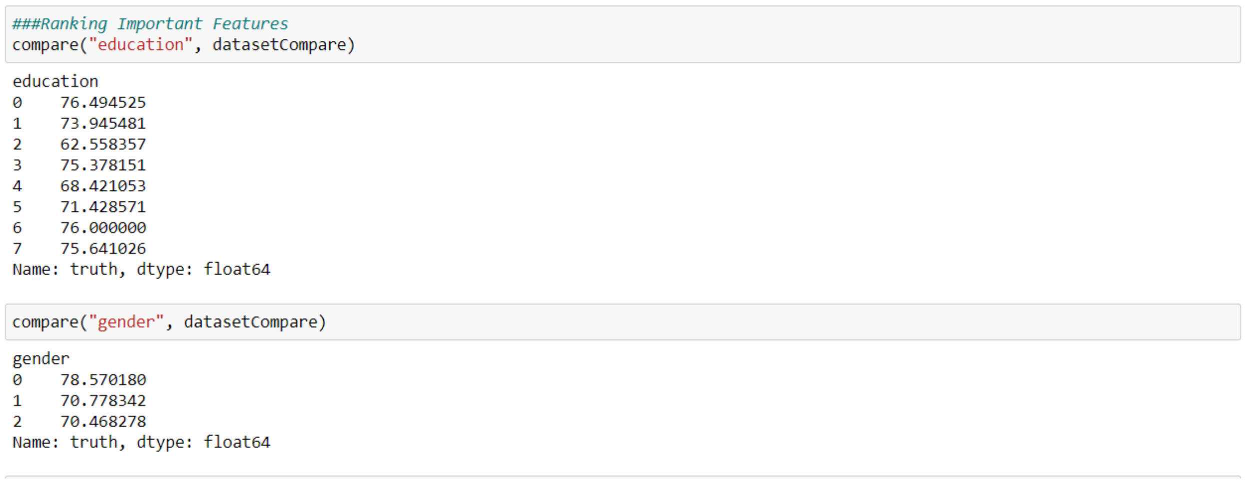
157943 rows × 119 columns

Additionally, for feature selection we have done comparative analysis using coding. In this study, we have performed feature ranking based on their click through rate and dropout prediction probability. Basically, we have compared the user clusters with their dropout/retention truth table values. Thereby, user's in cluster two are ranked with 79% importance in dropout prediction dataset.



```
In [4]: def compare(group, data):
            return data.groupby([group])['truth'].sum()*100/data.groupby([group])['truth'].count()
        compare("cluster_label", datasetCompare)

Out[4]: cluster_label
        0    80.365606
        1    31.707317
        2    79.134578
        3    70.878805
        4    35.266272
        Name: truth, dtype: float64
```

Furthermore, we have done the similar data exploratory analysis using user's education group, and user's age and user's gender.



```
###Ranking Important Features
compare("education", datasetCompare)

education
0    76.494525
1    73.945481
2    62.558357
3    75.378151
4    68.421053
5    71.428571
6    76.000000
7    75.641026
Name: truth, dtype: float64

compare("gender", datasetCompare)

gender
0    78.570180
1    70.778342
2    70.468278
Name: truth, dtype: float64
```

These data exploratory analysis helps us to visualize and select important features in the dataset and to understand the main causes and reasons behind user's dropout rates in MOOCS.

Let us first do experiment with simple baseline methods and compare their performances and suitability in log files for prediction problem.

5.2 Support Vector Machine

In this study, we will use the linear support vector machine classifier for binary classification prediction problem. By using SVM, we can predict the class of a new instance X with help of its decision function. (O'Reilly, 2019)

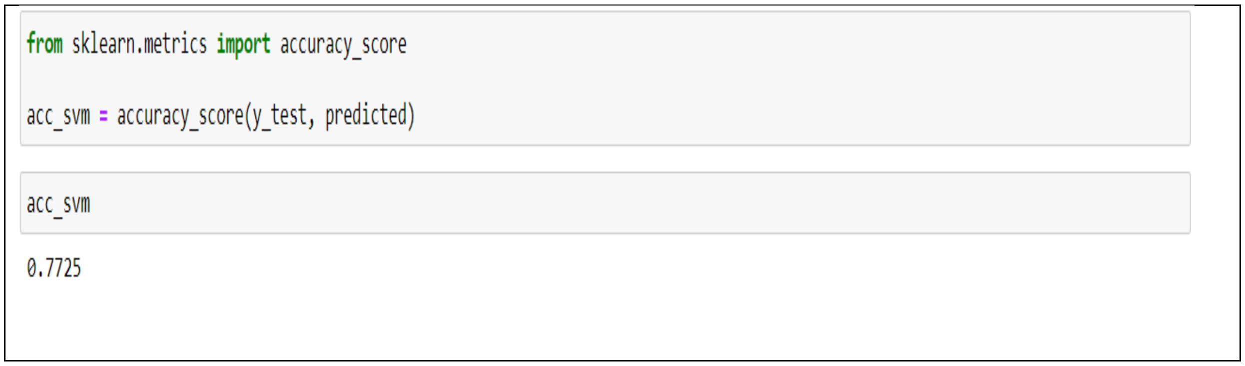
$$\hat{y} = \begin{cases} 0 \; if \; W^T x + b < 0, \\ 1 \; if \; W^T x + b \geq 0 \end{cases}$$

(O'Reilly, 2019) suggests the above-mentioned formula for linear SVM binary classification problem. The decision function is

$W^T x + b = \; w_1 x_1 + \; w_2 x_2 + \cdots \; w_n x_n + b$

The use of SVM is effective in high-dimensional spaces data but as the user learning activity log file data contains lot of features, we are required to avoid overfitting in choosing kernel functions. The SVM is a baseline method and for better performance results we should tune up its hyperparameters. Choosing the kernel is the most important parameter. Furthermore, the C parameter, penalty, and dual parameters are also considered while parameter tuning. For instance, setting C parameter will represent variety of values and it has a huge impact on the shapes of resulting regions for each class. We have used hinge loss function in SVM. During evaluation, we have obtained 77% accuracy using linear SVM approach. The accuracy rate is quite low as compared to other methodologies like DeepFM, CFIN which provides 90% accurate results. Thus, we have observed that using baseline methods in lengthy log files or let us say for prediction systems is not suitable. Because these methods ignore the implicit feature details in user learning activity which is why ignoring certain low-feature and high-feature interactions does not provide the best performance results.

Implementation Results in Keras Models using JupyterNotebook:



```
from sklearn.metrics import accuracy_score

acc_svm = accuracy_score(y_test, predicted)
```

```
acc_svm
```

```
0.7725
```

Prediction Method	Accuracy
SVM	0.7725

5.3 Decision Tree

Feature selection is very important aspect for prediction problems. The discrepancies in choosing the right features from user learning activity will fail to provide the accurate evaluation results for prediction methods. (Jing Chen, 2019) With decision tree algorithm, our goal is to achieve optimal results by doing feature selection with less data preparation. The main advantage of using decision trees for making prediction is that it requires very little data preparation, thus results in providing fast training. Additionally, one of its key features is that depending on decision tree structure it will assign weights to the selected features.

Decision Tree Implementation:

To be specific, firstly we will do feature extraction from the user learning activity and will use embedding layer for user and course context with learning activity as explained in CFIN methodology. After feature extraction, we will obtain feature matrix as follows:

$$x = [x_1, x_2, .., x_e]$$

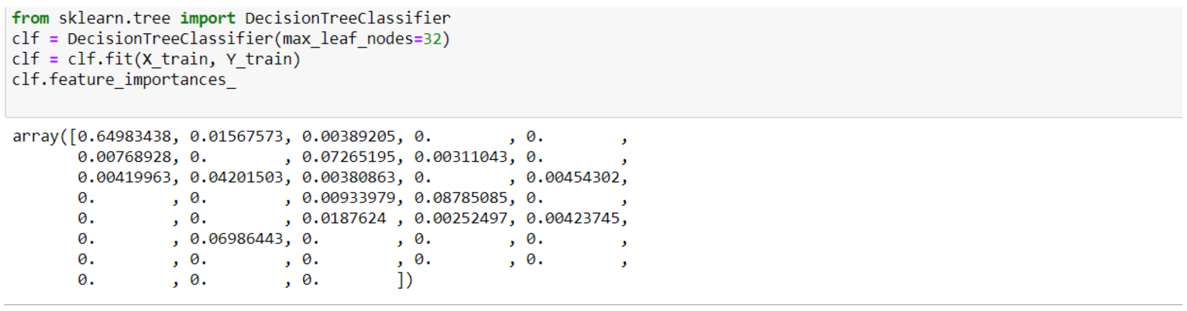
In the above equation, e denotes the number of users enrolled in the course.

$$y = [y_1, y_2, .., y_n]$$

Here, this equation is representing the label dropout or not dropout matrix (Jing Chen, 2019).

In the below picture, we have displayed the analysis results of feature selection rankings importance.

Additionally, note that Setting max_leaf_nodes == 32 results in accuracy with .828 but increasing max leaf nodes it will start decreasing the model accuracy.



```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(max_leaf_nodes=32)
clf = clf.fit(X_train, Y_train)
clf.feature_importances_
```

```
array([0.64983438, 0.01567573, 0.00389205, 0.        , 0.        ,
       0.00768928, 0.        , 0.07265195, 0.00311043, 0.        ,
       0.00419963, 0.04201503, 0.00380863, 0.        , 0.00454302,
       0.        , 0.        , 0.00933979, 0.08785085, 0.        ,
       0.        , 0.        , 0.0187624 , 0.00252497, 0.00423745,
       0.        , 0.06986443, 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        ])
```

Decision Layer:

After successfully performing feature extraction, the decision layer is used to do feature selection depending on the given decision tree maximum ratio.

According to (Jing Chen, 2019), D' is the output which contains only selected features as represented in the following equation:

$$D' => x_{inew} = [x_{i1}, x_{i2}, .., x_{ir}]$$

In this methodology, recursive partitioning is used in which D is partitioned into smaller subsets such as:

$$D_1, D_2, \dots, D_n$$

Recursive partition will not stop until reaching the specified stopping criterion where all classes will belong to one class.

Entropy Function suggested by (O'Reilly, 2019)**:**

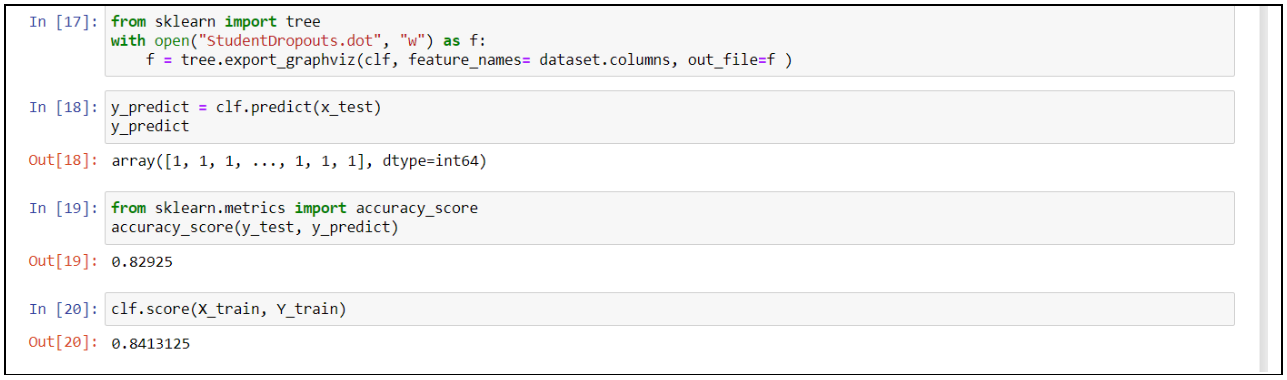
$$info(D) = -\sum_{k=1}^{m} p_k log_2 p_k$$

m : denotes the number of classes

p: probability that an instance belong to class k

Experimental Results:

With decision tree algorithm, we get accuracy score of 82% which is certainly higher than SVM prediction method, SVM provided 77% accuracy results.



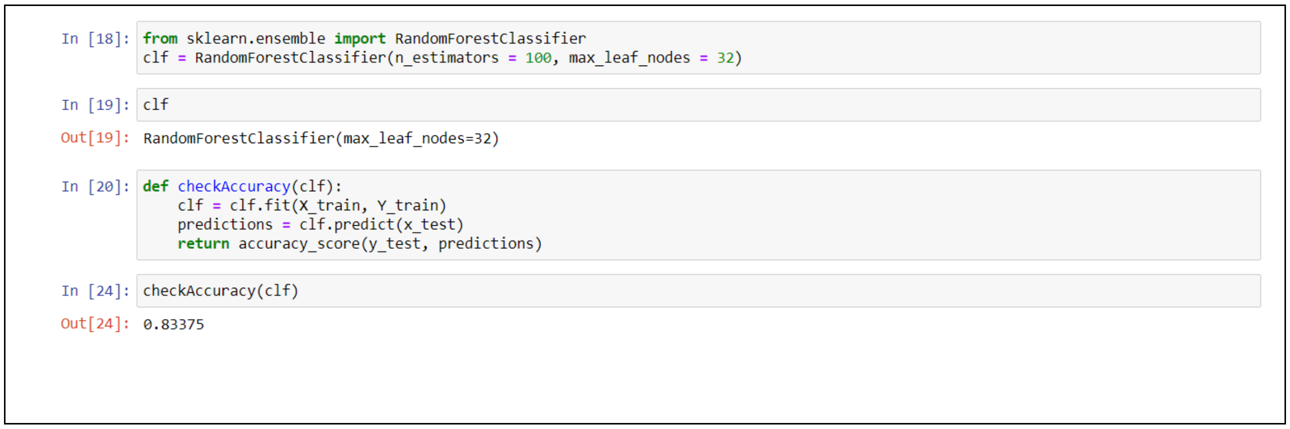
```
In [17]: from sklearn import tree
         with open("StudentDropouts.dot", "w") as f:
             f = tree.export_graphviz(clf, feature_names= dataset.columns, out_file=f )

In [18]: y_predict = clf.predict(x_test)
         y_predict

Out[18]: array([1, 1, 1, ..., 1, 1, 1], dtype=int64)

In [19]: from sklearn.metrics import accuracy_score
         accuracy_score(y_test, y_predict)

Out[19]: 0.82925

In [20]: clf.score(X_train, Y_train)

Out[20]: 0.8413125
```

Prediction Method	Accuracy
Decision Tree	0.82925

5.4 Random Forest

Random forest model approach is basically an ensemble modeling of decision tree algorithm. (O'Reilly, 2019) We will use keras implementation RandomForestClassifier class and set max_leaf_nodes to 32 and n_estimators to 500. The number of trees being chosen has a huge impact on classification model evaluation results. Different parameter tuning has been applied to assess optimal model results, such as setting tree number to 32, 100, 256 has been applied. So, increasing the number of trees is increasing the model accuracy results. Thus, it proves that RF with a greater number of trees provides more accurate and optimal results.

With Random Forest algorithm, we get accuracy score of 83% which is certainly higher than DT and SVM prediction methods which provided accuracy of 82% and 77% respectively.



```
In [18]: from sklearn.ensemble import RandomForestClassifier
         clf = RandomForestClassifier(n_estimators = 100, max_leaf_nodes = 32)

In [19]: clf

Out[19]: RandomForestClassifier(max_leaf_nodes=32)

In [20]: def checkAccuracy(clf):
             clf = clf.fit(X_train, Y_train)
             predictions = clf.predict(x_test)
             return accuracy_score(y_test, predictions)

In [24]: checkAccuracy(clf)

Out[24]: 0.83375
```

Prediction Method	Accuracy
Random Forest	0.83375

5.5 Voting Classifiers:

So, we have results for three different methods including SVM, DT and RF. With these baseline methods, we obtained accuracy score of 77%, 82% and 83% respectively. To achieve even better accuracy score rate, we propose to use voting classifier technique in which we will aggregate the predictions of all these classifiers and model will predict the class that gets the majority vote. (O'Reilly, 2019) One major advantage of using the ensemble voting methodology is that it uses different classifiers for predictions which is why different predictors makes different types of errors and resulting in better accuracy results. (O'Reilly, 2019)

Ensemble learning equation suggested by (E. Mulyani, 2019) using majority voting classifier is as follows:

$$\hat{y} = mode\{C_1(x), C_2(x), \ldots, C_m(x)\}$$

Where C denotes Classifiers, in this case we have three classifiers such as SVM, DT and RF.
The ensemble learning method can be basically summarized as if two of classifiers out of three provides dropout value as true then the final prediction would be considered as dropout.

5.6 Model Evaluation:

In the evaluation results for ensemble approach over current state-of-the-art baseline models has received much higher accuracy results comparatively single classifiers. That is why ensemble learning approach is encouraged to get more accurate results for prediction methods.

Evaluation Results	Proposed methods			
	EL	SVM	DT	RF
Accuracy Score	0.84	0.7725	0.82925	0.83375

Chapter 6: Performance Evaluation

6.1 CFIN Evaluation Metrics:

6.1.1 The ROC Curve:

The ROC curve score for the CFIN model is 0.8663. Which is still equivalent to CFIN model proposed by (Feng, 2019). But with this proposed approach, time elapse is less than previous approach with higher model output efficiency.

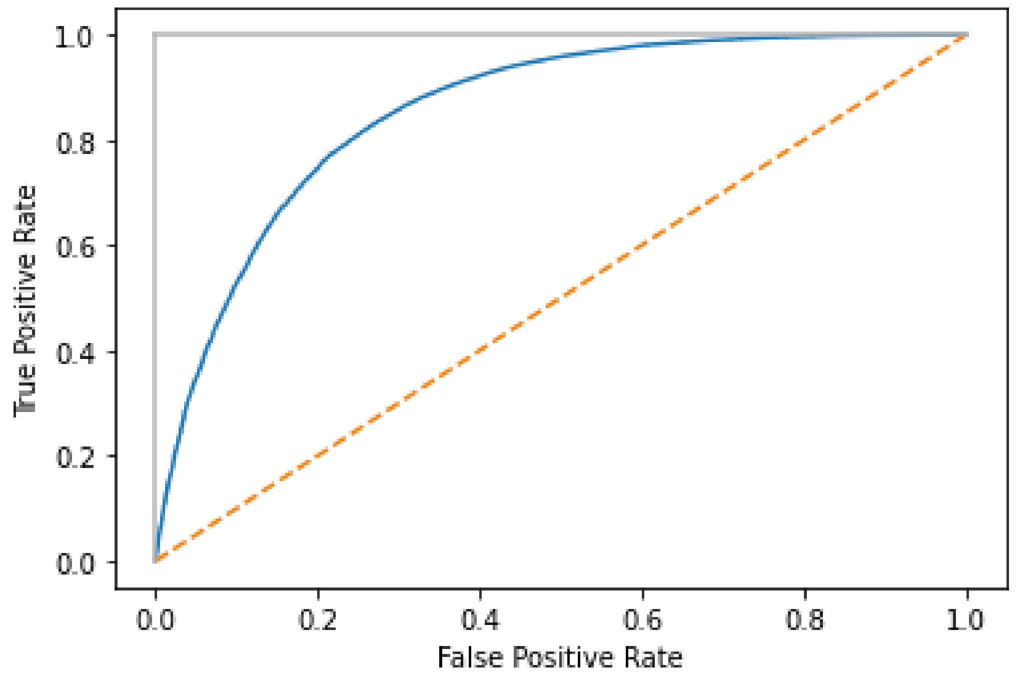


Figure 11CFIN Model: ROC Curve

6.1.2 F1-Score:

The F1-score is used to combine and measure the precision and recall in a single metric. The F1-score is known as the harmonic mean of recall and precision. Thus, harmonic mean gives more weight to low-values.

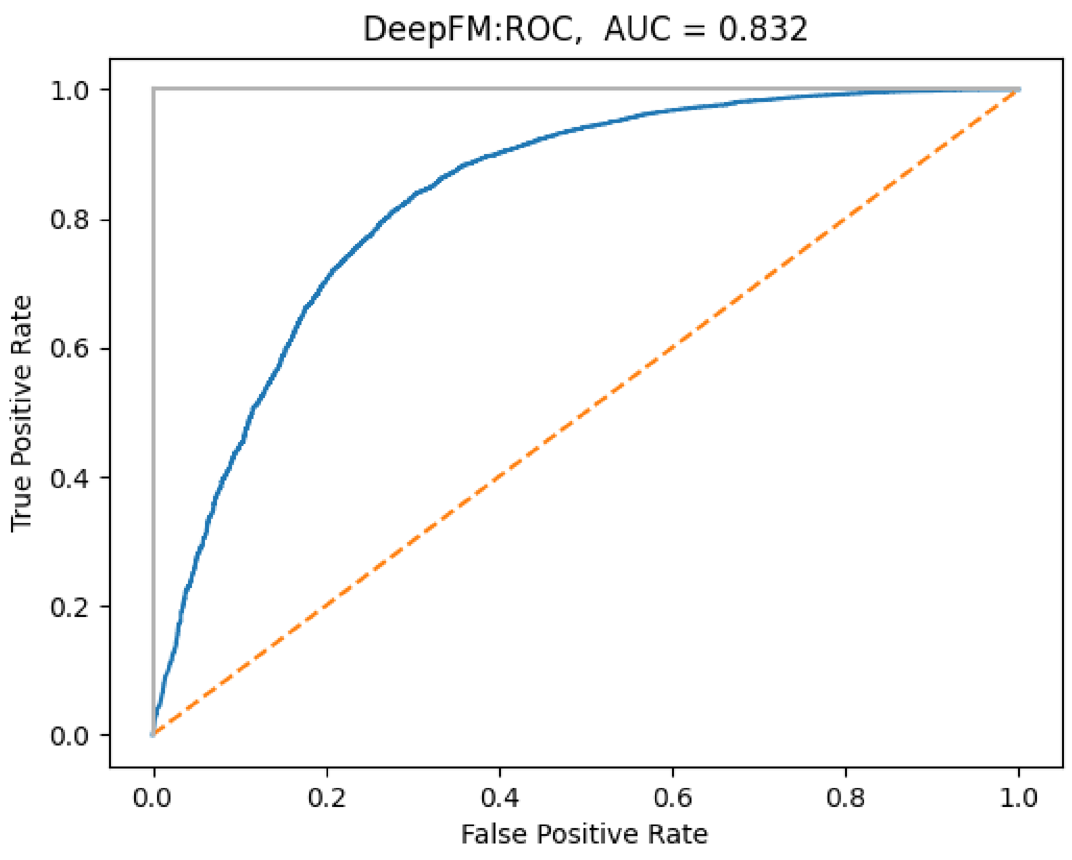
In the CFIN model, we have trained our model to predict students who are at higher risk of dropout andstudents with lowest rates of being dropout from a course. The proposed model achieves F1-score equals to 92 percent. (Note: in the code example sample dataset has been used which is why the performance results can vary a bit)

AUC Score	F1- Score
.8663	.9239

6.2 DeepFM Evaluation Metrics:

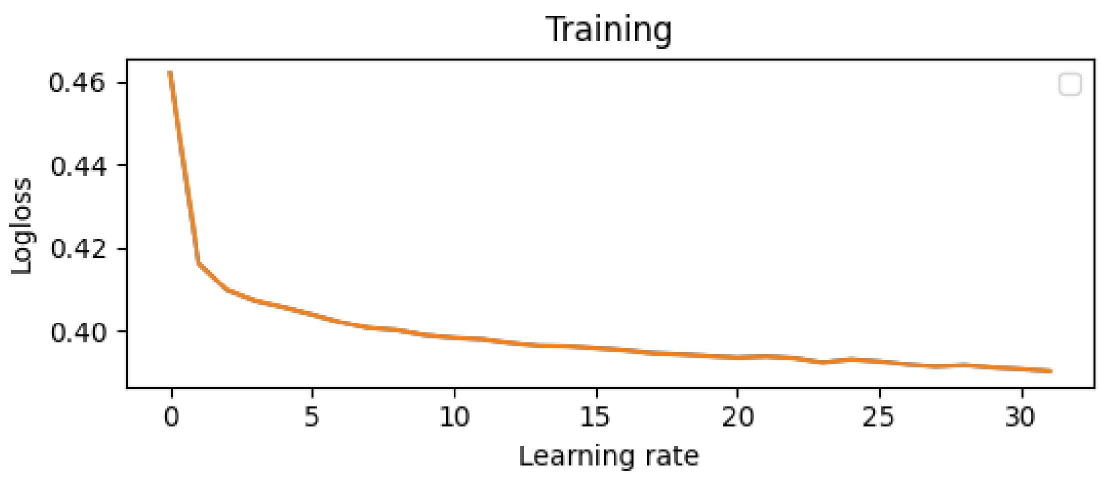
6.2.1 The ROC Curve and logloss function:

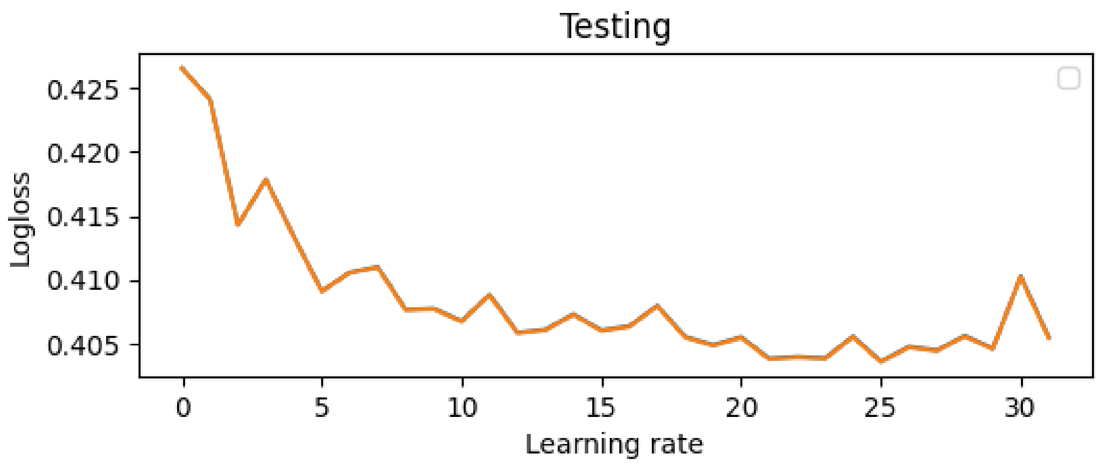
In this ROC curve, it plots the false positive rates against the true positive rates for all possible thresholds. We have also computed the AUC score which is AUC = 0.832 and its logloss = 0.3919.



AUC = 0.832	**Logloss = 0.3919**

Here, is the practical implementation of logloss for user learning activity embedded with user and course context using DeepFM method. The logloss score for training dataset is around 0.46 maximum while for testing dataset the logloss score lies around 0.42.





Learning Rate: If the learning rate is too small or too large it would not provide the optimal value. To accuratly estimate the optimal value choosing an appropriate learning rate would be very effective approach.

Learning rate = 0.0001	Logloss = 0.3953	AUC: Accuracy = 0.8322

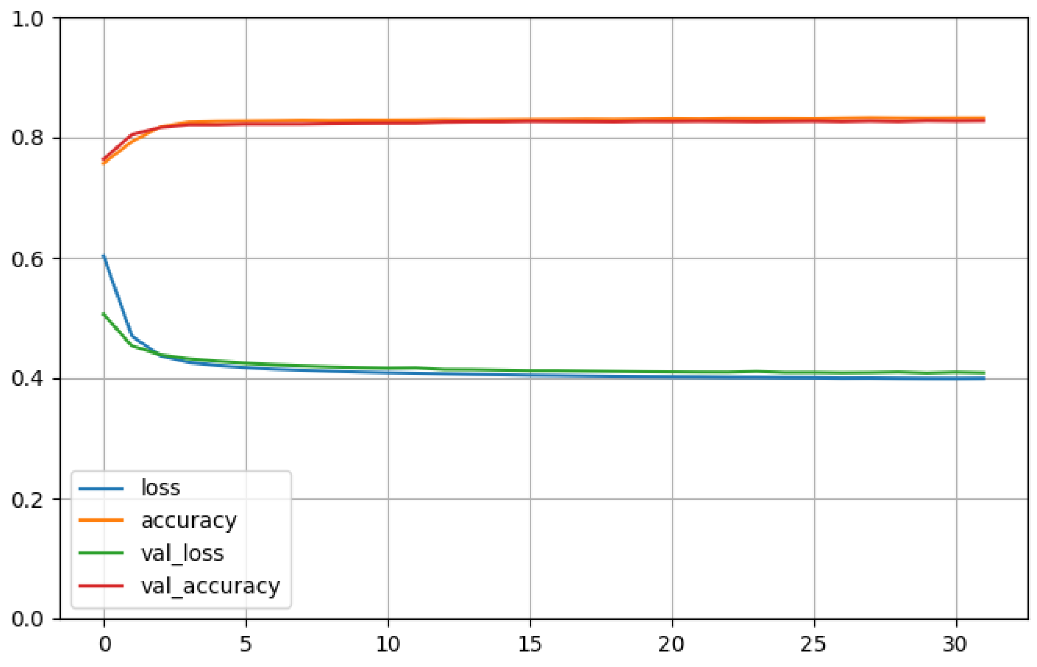


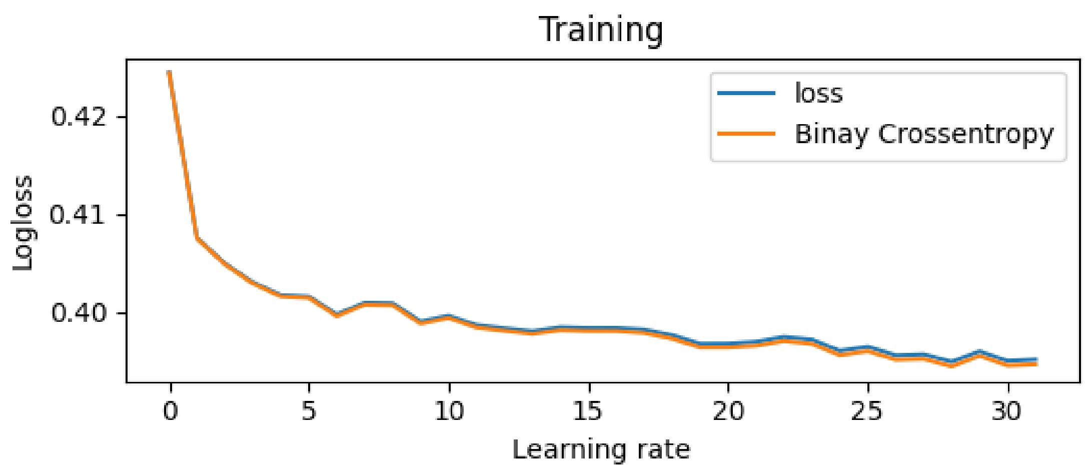
Figure 0-2Learning Curve: The mean training logloss and accuracy has been measured over each epoch and also the mean of validation logloss and validation accuracy is measured.

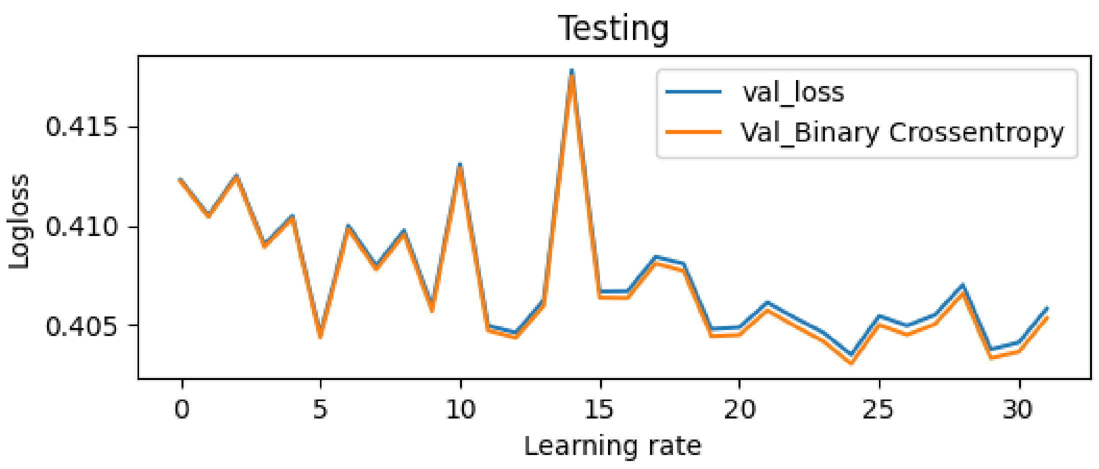
Learning rate = 0.001	Logloss = 0.392	AUC: Accuracy = 0.8411

By decreasing the learning rate value from 0.0001 to 0.001, we get reduced logloss function and increased accuracy value to 84 percent. While setting learning rate to 0.01 will fail to provide optimal output values.

Learning rate = 0.01	Logloss = 0.3918	AUC: Accuracy = 0.83

It has decreased the loss value, but it also reduced model accuracy score from 84 to 83 percent. By adjusting the parameter weights and values, the model AUC score improves to 83% with logloss as 0.3911. Albeit the logloss value is still very same with default parameter values but these changes are still improving the model accuracy in a smaller range.





6.3 Ensemble Learning and Baseline Methods Comparison:

In the evaluation results for ensemble approach over current state-of-the-art baseline models has received much higher accuracy results comparatively single classifiers. That is why ensemble learning approach is encouraged to get more accurate results for prediction methods.

Evaluation Results	Proposed methods			
	EL	SVM	DT	RF
Accuracy Score	0.84	0.7725	0.82925	0.83375

6.4 Comparative Analysis:

So far, we have our experimental results for single classifier baseline models and ensemble learning, and DeepFM model results for CTR prediction method and CFIN model performance results. In this section, we will compare those modelling approaches with their evaluation metric results and choose the best modelling technique proposed for the dropout prediction problem.

Overall Results on XuetangX dataset		
Methods	**Evaluation Metrics**	
	AUC-ROC Score	**F1-Score**
SVM	0.7725	87.35
DT	0.82925	87.78
RF	0.83375	88.96
EL	0.84	88.99
DeepFM	0.85	89.92
CFIN	0.86	90.95

For baseline models, we have used all contextual features including user, course, and user learning activity in enrolled courses. Overall, CFIN model achieves the best performance score on both AUC and F1 scores. While the DeepFM model has been proven better approach than baseline methods and ensemble techniques, but its performance score is little less comparatively CFIN modelling approach.

Chapter 7: Conclusion

In this study, I investigated two variants of the CFIN Model with a personalized attention mechanism to model and predict user dropout behaviors in MOOCs. A dataset from XuetangX MOOC platform was acquired, and a systematic study for dropouts' problems in MOOCs was conducted. We have proposed changes in the original CFIN model proposed by (Feng, 2019) by separating user encoder and course encoder models. The core of our approach is contextual information such as user representation model, course representation model, and user click predictor model. We used user-context model to learn user representation by their clicked activity in an enrolled course such as user forum activities, videos activities, assignments activities. And we used the course-context module to learn course representation in user's learning activity based on course category and type. The proposed CFIN model has utilized the context smoothing technique for both user and course representation models to smooth feature values and used CNN with personalized attention mechanism to combine user and course information into the modeling framework. Additionally, the same course can have different dropout rates for different user groups, thus it is proposed to build a personalized attention network that exploits the embedding of a user ID and embedding of course ID to generate the query vector for the course and learning-activity level attentions. Experiments on the dataset show that the two proposed variants have similar performance with the CFIN model, predicting dropouts with a 93% accuracy, and it achieves better performance than baseline methods including SVM, RF, DT, and ensemble modeling techniques and the DeepFM method.

Another notable contribution in this study, is the use of the DeepFM method for student's dropout prediction problem. After doing systimatical study, I proposed a personalized student retention method using a deep factorization machine (DeepFM) to learn implicit interactions behind user click behaviors on MOOCs dataset. The proposed method, DeepFM, integrates the power of factorization machines (FM) and deep neural network (DNN) for low-to-high level feature learning into a new neural network architecture. With DeepFM parameter sharing strategy, the relationship between low order and high order feature interactions are learned from log data and can be trained end-to-end efficiently without any feature engineering and click rate prediction model is constructed. Finally, based on the predicted CTR, the area under the curve (AUC) and Logloss of the DeepFM method are 0.8423 and 0.3915 on XuetangX dataset for students dropout prediction.

7.1 Contribution to the CFIN method proposed by (Feng, 2019)

We have proposed to separate the course context and user context module and later embed the modules in attention-based CNN architecture. There are three major modules: User Encoder, Course Encoder, and Click Predictor. In the user encoder module, it aims to learn user profile representation based on the number of times the user clicked on a course activity such as watch/stop/jump course video, forum activities. In the course encoder, the goal is to learn course representation. While in the Click Predictor module, it is used to estimate the click through rate of a user learning activity. In the model architecture figure, it has been displayed that we are using attention-based CNN mechanism for user and course modules to predict student's dropout rates in MOOCs. In this model, CNN with attention mechanism is used in course-context and user-context to extract the course and user features. Additionally, the strength of using attention mechanism in CNN model is to find latent factors. Finally, the proposed model will obtain latent features both at personal and statistical levels. (J. Wang, 2020)

7.2 Results discussion

So far, we have used baseline modeling techniques for students dropout prediction problem including SVM, DT and RF that provides AUC score as 77%, 82% and 83%, respectively. However, we have noticed that baseline models are not sufficient for huge dataset files, in this case user tracking log files at XuetanX dataset which records user's clickstream data in an enrolled course. Another demerit of using baseline approaches on CTR based dataset is loosing the implicit feature interactions. Thus, to deal with CTR prediction based data, we have employed the DeepFM method which provides 84% accuracy and learns end-to-end feature interactions in the model that means learning feature interactions from low-to-high feature relationships. Overall, CFIN has been proven the best approach to be used for MOOCs Dropout prediction problem because of its impact on learning contextual information with attention mechanism. CFIN has provided an AUC score of 86% on XuetangX dataset and F1-score of 93%. Thus, CFIN has provided best performance results over the DeepFM and ensemble modeling approaches.

7.3 Future Work

Due to time constraints and project scope, I could not do experiments using the recurrent neural network approaches with LSTM. However, I am eager to do further research work to improve accuracy score for MOOC's dropout prediction problem by utilizing different methodologies. The code for this project will be available online through github and other researchers are also encouraged to come with any improvements in modelling CFIN approach. Another issue, I have noticed while implementing DeepFM is that it provides higher performance results for ROC, precision, and F1-score and outperformed ensemble learning and other baseline models. However, it has been observed that the recall score for the DeepFM approach is little less than the baseline models. To improve recall score, I am also opened to do further research work to increase model performance results.

References

(n.d.). Retrieved from moocdata.cn: http://moocdata.cn/data/user-activity

Ahmed A. Mubarak, H. C. (2017). Prediction of students' early dropout based on their interaction logs in online learning environment. *DOI: 10.1080/10494820.2020.1727529*.

B. Hong, Z. W. (2017). Discovering learning behavior patterns to predict dropout in MOOC. *2017 12th International Conference on Computer Science and Education (ICCSE).*

deepctr.models.deepfm.html. (n.d.). Retrieved from deepctr-doc.readthedocs.io: https://deepctr-doc.readthedocs.io/en/latest/deepctr.models.deepfm.html

E. Mulyani, I. H. (2019). Dropout Prediction Optimization through SMOTE and Ensemble Learning. *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI).*

Feng, W. T. (2019). Understanding Dropouts in MOOCs. *Proceedings of the AAAI Conference on Artificial Intelligence.* AAAI.

Huifeng Guo, R. T. (2017). DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17).* arXiv.

J Xu, Z. H. (2021). Personalized Product Recommendation Method for Analyzing User Behavior Using DeepFM. *Journal of Information Processing Systems*.

J. Huang, X. Z. (2019). CoStock: A DeepFM Model for Stock Market Prediction with Attentional Embeddings. *2019 IEEE International Conference on Big Data (Big Data).* IEEE.

J. Wang, H. X. (2020). Attention-Based CNN for Personalized Course Recommendations for MOOC Learners. *2020 International Symposium on Educational Technology (ISET), 2020.* IEEE.

Jing Chen, J. F. (2019). MOOC Dropout Prediction Using a Hybrid Algorithm Based on Decision Tree and extreme learning machine. *Hindawi, Mathematical Problems in Engineering Volume 2019*.

MOOCCube. (n.d.). Retrieved from moocdata.cn: http://moocdata.cn/data/MOOCCube

O'Reilly. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition.* O'Reilly Media, Inc.

Wu, C. e. (2019). NPA: Neural News Recommendation with Personalized Attention. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining .*

wzfhaha. (n.d.). Retrieved from https://github.com/: https://github.com/wzfhaha/dropout_prediction

Y. Zhang, L. C. (2020). MOOCs Dropout Prediction Based on Hybrid Deep Neural Network. *2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC).*

Yeung, M. F. (2015). Temporal Models for Predicting Student Dropout in Massive Open Online Courses. *2015 IEEE International Conference on Data Mining Workshop (ICDMW).*

Z. Yu, S. U. (2021). Research on Disease Prediction Based on Improved DeepFM and IoMT. *IEEE Access*.

Zhemin Liu, F. X. (2018). Predicting Learning Status in MOOCs using LSTM.

Appendices

Appendix 1 GitHub Repository Link

https://github.com/zunairazaman2021/MOOCSDropoutsPrediction

Appendix 2 Code Developed for this Project

DeepFM model Code:

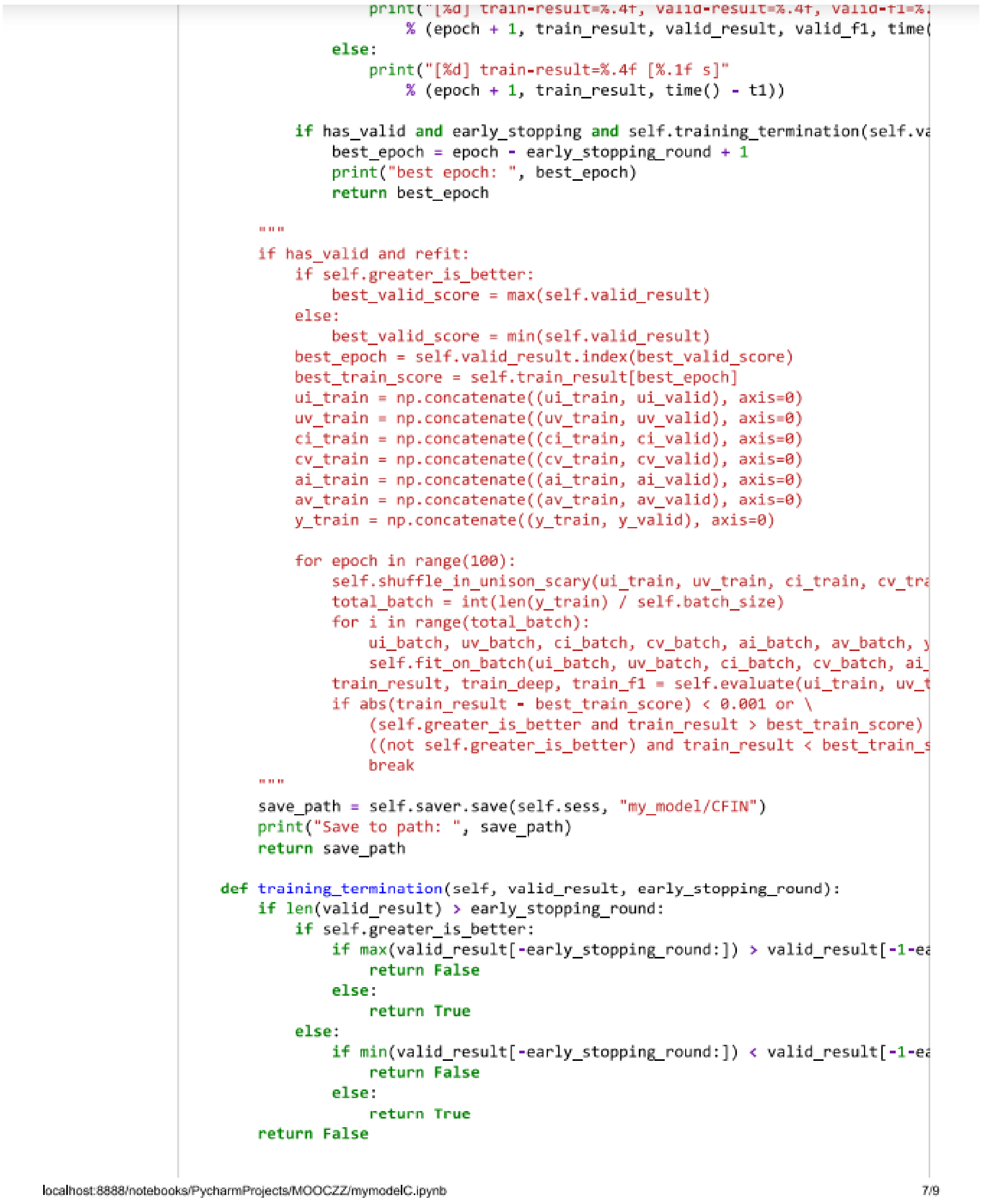
```
import pandas as pd
from sklearn.metrics import log_loss, roc_curve, roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler

from deepctr.models import DeepFM
from deepctr.feature_column import SparseFeat, DenseFeat, get_feature_names
from matplotlib import pyplot
from tensorflow.keras.utils import plot_model
from keras.optimizers import SGD
if __name__ == "__main__":
    data = pd.read_csv('./DeepFMDummy.csv')
    data.head()
    print(data.head())
    sparse_features = [ 'gender', 'education',  'cluster_label', 'course_categor
y']
    dense_features = ['all#count', 'session#count', 'seek_video#num', 'play_vide
o#num',
     'pause_video#num', 'stop_video#num', 'load_video#num','age']

    dense_features = ['all#count', 'session#count', 'seek_video#num', 'play_vide
o#num',
        'pause_video#num', 'stop_video#num', 'load_video#num',    'problem_get#nu
m',
        'problem_check#num', 'problem_save#num','reset_problem#num', 'problem_ch
eck_correct#num',
        'problem_check_incorrect#num', 'create_thread#num', 'create_comment#num'
, 'delete_thread#num',
        'delete_comment#num','click_info#num', 'click_courseware#num', 'click_ab
out#num','click_forum#num',
        'click_progress#num', 'close_courseware#num', 'age']

    target = ['truth']

    # 1.Label Encoding for sparse features,and do simple Transformation for dens
e features
    for feat in sparse_features:
        lbe = LabelEncoder()
        data[feat] = lbe.fit_transform(data[feat])
    mms = MinMaxScaler(feature_range=(0, 1))
    data[dense_features] = mms.fit_transform(data[dense_features])

    fixlen_feature_columns = [SparseFeat(feat, vocabulary_size=data[feat].max()
+ 1,embedding_dim=32 )
                           for i,feat in enumerate(sparse_features)] + [DenseFea
t(feat, 1,)
                          for feat in dense_features]

    dnn_feature_columns = fixlen_feature_columns
    linear_feature_columns = fixlen_feature_columns
    feature_names = get_feature_names(linear_feature_columns + dnn_feature_colum
ns)

    train, test = train_test_split(data, test_size=0.2, random_state=2020)
    train_model_input = {name:train[name] for name in feature_names}
    test_model_input = {name:test[name] for name in feature_names}


    # 4.Define Model,train,predict and evaluate
    model = DeepFM(linear_feature_columns, dnn_feature_columns, task='binary')
    model.compile("adam", "binary_crossentropy",
                metrics=['binary_crossentropy'], )

    history = model.fit(train_model_input, train[target].values,
                        batch_size=256, epochs=32, verbose=2, validation_split=0
.2, )
    pred_ans = model.predict(test_model_input, batch_size=256)
    print("test LogLoss", round(log_loss(test[target].values, pred_ans), 4))
    print("test AUC", round(roc_auc_score(test[target].values, pred_ans), 4))

    pd.DataFrame(history.history).plot(figsize=(8, 5))
    pyplot.grid(True)
    pyplot.gca().set_ylim(0,1)
    pyplot.show()


    model.summary()

    plot_model(model, to_file='C:/Users/zunai/OneDrive/Desktop/FYP_MOOCS/Writing 
Thesis/submodelDFM.jpg')
    score1 = pred_ans
    f, t , threshold = roc_curve(test[target].values, score1)
    pyplot.title('DeepFM:ROC')
    pyplot.plot(f, t)
    pyplot.plot([0, 1], ls="--")
    pyplot.plot([0, 0], [1, 0] , c=".7"), pyplot.plot([1, 1] , c=".7")
    pyplot.ylabel('True Positive Rate')
    pyplot.xlabel('False Positive Rate')
    pyplot.show()






    pyplot.subplot(211)
    pyplot.title('Training')
    pyplot.plot(history.history['loss'])
    pyplot.plot(history.history['binary_crossentropy'])
    pyplot.xlabel("Learning rate")
    pyplot.ylabel('Logloss')
    pyplot.legend()
    pyplot.show()

    # plot accuracy during training
    pyplot.subplot(212)
    pyplot.title('Testing')
    pyplot.plot(history.history['val_loss'])
    pyplot.plot(history.history['val_binary_crossentropy'])
    pyplot.xlabel("Learning rate")
    pyplot.ylabel('Logloss')
    pyplot.legend()
    pyplot.show()
```

```
import pandas as pd
from sklearn.metrics import log_loss, roc_curve, roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from tensorflow import keras

from deepctr.models import DeepFM
from deepctr.feature_column import SparseFeat, DenseFeat, get_feature_names
from matplotlib import pyplot
from tensorflow.keras.utils import plot_model
from keras.layers import Dense
if __name__ == "__main__":
    data = pd.read_csv('./DeepFMDummy.csv')
    data.head()
    print(data.head())
    sparse_features = [ 'gender', 'education',  'cluster_label', 'course_categor
y']
   # dense_features = ['all#count', 'session#count', 'seek_video#num', 'play_vid
eo#num',
   #  'pause_video#num', 'stop_video#num', 'load_video#num','age']

    dense_features = ['all#count', 'session#count', 'seek_video#num', 'play_vide
o#num',
        'pause_video#num', 'stop_video#num', 'load_video#num',    'problem_get#nu
m',
     'problem_check#num', 'problem_save#num','reset_problem#num', 'problem_check
_correct#num',
     'problem_check_incorrect#num', 'create_thread#num', 'create_comment#num', '
delete_thread#num',
     'delete_comment#num','click_info#num', 'click_courseware#num', 'click_about
#num','click_forum#num',
     'click_progress#num', 'close_courseware#num', 'age']

    target = ['truth']

    # 1.Label Encoding for sparse features,and do simple Transformation for dens
e features
    for feat in sparse_features:
        lbe = LabelEncoder()
        data[feat] = lbe.fit_transform(data[feat])
    mms = MinMaxScaler(feature_range=(0, 1))
    data[dense_features] = mms.fit_transform(data[dense_features])

    fixlen_feature_columns = [SparseFeat(feat, vocabulary_size=data[feat].max()
+ 1,embedding_dim=32 )
```

```
                           for i,feat in enumerate(sparse_features)] + [DenseFea
t(feat, 1,)
                          for feat in dense_features]

    dnn_feature_columns = fixlen_feature_columns
    linear_feature_columns = fixlen_feature_columns
    feature_names = get_feature_names(linear_feature_columns + dnn_feature_colum
ns)

    train, test = train_test_split(data, test_size=0.2, random_state=2020)
    train_model_input = {name:train[name] for name in feature_names}
    test_model_input = {name:test[name] for name in feature_names}


    # 4.Define Model,train,predict and evaluate
    model = DeepFM(linear_feature_columns, dnn_feature_columns, l2_reg_linear =
0.00001, l2_reg_embedding=0.00001 ,
            dnn_hidden_units=(128, 128), dnn_dropout=0.1, dnn_activation='relu',
 task='binary')
    opt = keras.optimizers.Adam(learning_rate=0.01)
    #model.compile("adam", "binary_crossentropy",
     #              metrics=['binary_crossentropy'], )

    model.compile(loss = 'binary_crossentropy', optimizer = opt, metrics=['binar
y_crossentropy'], )
    history = model.fit(train_model_input, train[target].values,
                        batch_size=256, epochs=32, verbose=2, validation_split=0
.2, )
    pred_ans = model.predict(test_model_input, batch_size=256)
    print("test LogLoss", round(log_loss(test[target].values, pred_ans), 4))
    print("test AUC", round(roc_auc_score(test[target].values, pred_ans), 4))


    pyplot.subplot(211)
    pyplot.title('Training')
    pyplot.plot(history.history['loss'], label='loss')
    pyplot.plot(history.history['binary_crossentropy'], label='Binay Crossentrop
y')
    pyplot.xlabel("Learning rate")
    pyplot.ylabel('Logloss')
    pyplot.legend()
    pyplot.show()

    # plot accuracy during training
    pyplot.subplot(212)
    pyplot.title('Testing')
    pyplot.plot(history.history['val_loss'], label= 'val_loss')
    pyplot.plot(history.history['val_binary_crossentropy'], label='Val_Binary Cr
ossentropy')
    pyplot.xlabel("Learning rate")
    pyplot.ylabel('Logloss')
    pyplot.legend()
    pyplot.show()
```

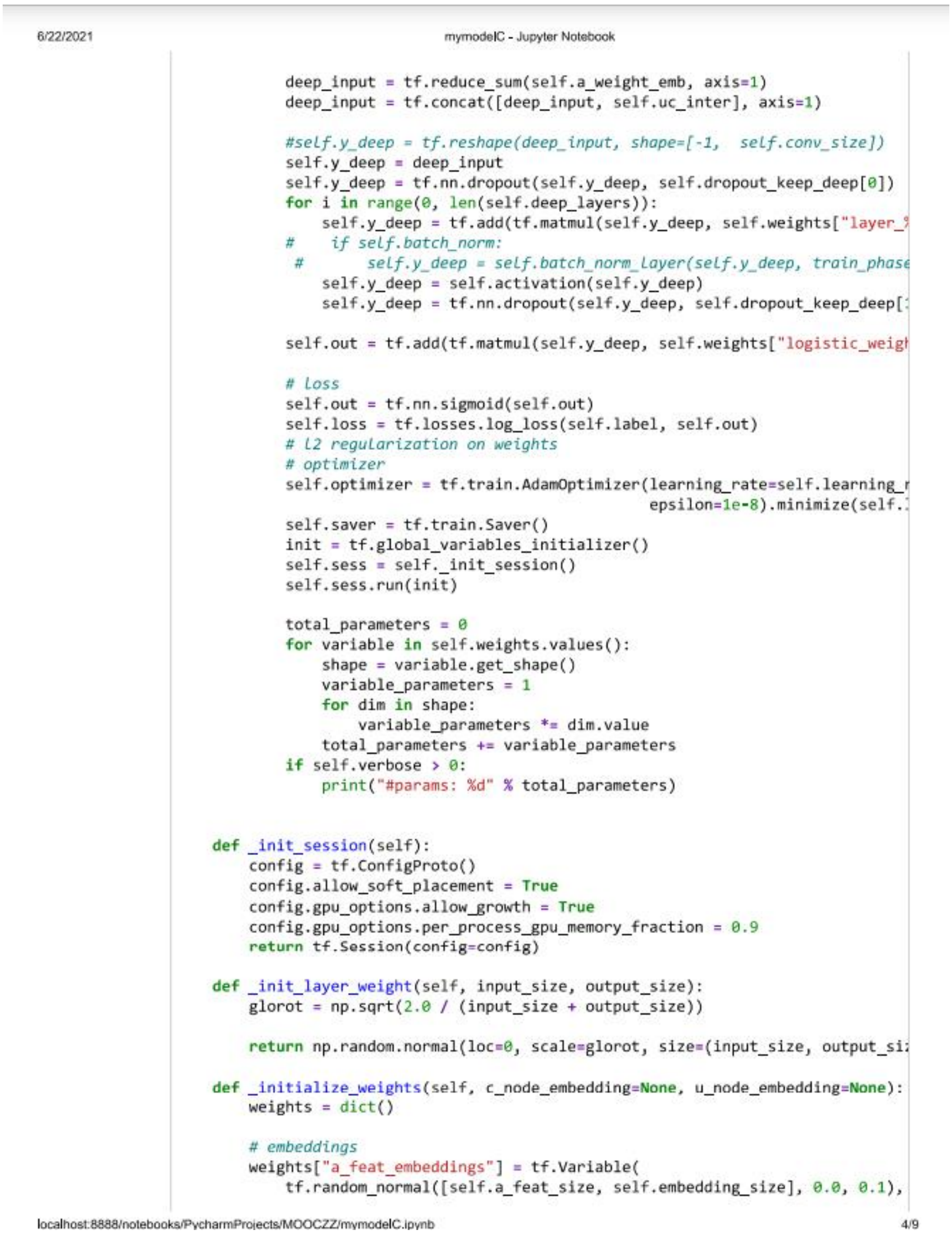
CFIN Model Code:

```
    def get_feats(self, ui,uv,ci,cv, ai, av, fname):
        dummy_y = [[1]] *len(ui)
        feed_dict = {self.u_feat_index: ui,
                     self.u_feat_value: uv,
                     self.c_feat_index: ci,
                     self.c_feat_value: cv,
                     self.a_feat_index: ai,
                     self.a_feat_value: av,
                     self.dropout_keep_deep: [1.0] * len(self.dropout_deep),
                     self.dropout_keep_attn: [1.0] * len(self.dropout_attn),
                     self.label: dummy_y,
                     self.train_phase: False}

        y_deep, y_rate= self.sess.run([self.y_deep, self.out], feed_dict = feed_c
        return y_deep,y_rate
    def get_attn(self, ui, uv, ci, cv, ai, av):
        dummy_y = [1] * len(ui)
        batch_index = 0
        ui_batch, uv_batch, ci_batch, cv_batch, ai_batch, av_batch ,y_batch = se]
        attn_weight = None
        uca_weight = None
        while len(ui_batch) > 0:
            num_batch = len(y_batch)
            feed_dict = {self.u_feat_index: ui_batch,
                         self.u_feat_value: uv_batch,
                         self.c_feat_index: ci_batch,
                         self.c_feat_value: cv_batch,
                         self.a_feat_index: ai_batch,
                         self.a_feat_value: av_batch,
                         self.dropout_keep_deep: [1.0] * len(self.dropout_deep),
                         self.dropout_keep_attn: [1.0] * len(self.dropout_attn),
                         self.label: y_batch,
                         self.train_phase: False}
            attn, uca_inter = self.sess.run([self.attn_w,self.uca_inter], feed_di
            if batch_index == 0:
                attn_weight = np.reshape(attn, (num_batch,-1))
                uca_weight = np.reshape(uca_inter, (num_batch, -1, self.embedding
            else:
                attn_weight = np.concatenate((attn_weight, np.reshape(attn, (num_
                uca_weight = np.concatenate((uca_weight, np.reshape(uca_inter, (r
            batch_index += 1

            ui_batch, uv_batch, ci_batch, cv_batch, ai_batch, av_batch ,y_batch =
        return attn_weight, uca_weight


    def predict(self, ui, uv, ci, cv, ai, av):
        dummy_y = [1] * len(ui)
        batch_index = 0
        ui_batch, uv_batch, ci_batch, cv_batch, ai_batch, av_batch ,y_batch = se]
        y_pred = None
        while len(ui_batch) > 0:
            num_batch = len(y_batch)
            feed_dict = {self.u_feat_index: ui_batch,
                         self.u_feat_value: uv_batch,
                         self.c_feat_index: ci_batch,
                         self.c_feat_value: cv_batch,
```



```
                    print("[%d] train-result=%.4f, valid-result=%.4f, valid-f1=%.
                        % (epoch + 1, train_result, valid_result, valid_f1, time(
                else:
                    print("[%d] train-result=%.4f [%.1f s]"
                        % (epoch + 1, train_result, time() - t1))

            if has_valid and early_stopping and self.training_termination(self.va
                best_epoch = epoch - early_stopping_round + 1
                print("best epoch: ", best_epoch)
                return best_epoch

        """
        if has_valid and refit:
            if self.greater_is_better:
                best_valid_score = max(self.valid_result)
            else:
                best_valid_score = min(self.valid_result)
            best_epoch = self.valid_result.index(best_valid_score)
            best_train_score = self.train_result[best_epoch]
            ui_train = np.concatenate((ui_train, ui_valid), axis=0)
            uv_train = np.concatenate((uv_train, uv_valid), axis=0)
            ci_train = np.concatenate((ci_train, ci_valid), axis=0)
            cv_train = np.concatenate((cv_train, cv_valid), axis=0)
            ai_train = np.concatenate((ai_train, ai_valid), axis=0)
            av_train = np.concatenate((av_train, av_valid), axis=0)
            y_train = np.concatenate((y_train, y_valid), axis=0)

            for epoch in range(100):
                self.shuffle_in_unison_scary(ui_train, uv_train, ci_train, cv_tra
                total_batch = int(len(y_train) / self.batch_size)
                for i in range(total_batch):
                    ui_batch, uv_batch, ci_batch, cv_batch, ai_batch, av_batch, y
                    self.fit_on_batch(ui_batch, uv_batch, ci_batch, cv_batch, ai_
                train_result, train_deep, train_f1 = self.evaluate(ui_train, uv_t
                if abs(train_result - best_train_score) < 0.001 or \
                    (self.greater_is_better and train_result > best_train_score)
                    ((not self.greater_is_better) and train_result < best_train_s
                    break
        """
        save_path = self.saver.save(self.sess, "my_model/CFIN")
        print("Save to path: ", save_path)
        return save_path

    def training_termination(self, valid_result, early_stopping_round):
        if len(valid_result) > early_stopping_round:
            if self.greater_is_better:
                if max(valid_result[-early_stopping_round:]) > valid_result[-1-ea
                    return False
                else:
                    return True
            else:
                if min(valid_result[-early_stopping_round:]) < valid_result[-1-ea
                    return False
                else:
                    return True
        return False
        end = (index+1) * batch_size
        end = end if end < len(y) else len(y)
        return ui[start:end], uv[start:end], ci[start:end], cv[start:end], ai[sta

    def shuffle_in_unison_scary(self, a, b, c, d,e,f,g):
        rng_state = np.random.get_state()
        np.random.shuffle(a)
        np.random.set_state(rng_state)
        np.random.shuffle(b)
        np.random.set_state(rng_state)
        np.random.shuffle(c)
        np.random.set_state(rng_state)
        np.random.shuffle(d)
        np.random.set_state(rng_state)
        np.random.shuffle(e)
        np.random.set_state(rng_state)
        np.random.shuffle(f)
        np.random.set_state(rng_state)
        np.random.shuffle(g)

    def fit_on_batch(self, ui, uv, ci, cv, ai, av, y):
        feed_dict = {self.u_feat_index: ui,
                     self.u_feat_value: uv,
                     self.c_feat_index: ci,
                     self.c_feat_value: cv,
                     self.a_feat_index: ai,
                     self.a_feat_value: av,
                     self.dropout_keep_deep: self.dropout_deep,
                     self.dropout_keep_attn: self.dropout_attn,
                     self.label: y,
                     self.train_phase: True}
        loss, opt = self.sess.run((self.loss, self.optimizer), feed_dict=feed_dic
        return loss

    def fit(self, ui_train, uv_train, ci_train, cv_train, ai_train, av_train, y_t
            ui_valid=None, uv_valid=None, ci_valid = None, cv_valid = None, ai_va
            early_stopping = True, early_stopping_round = 5, max_epoch =None):
        has_valid = uv_valid is not None
        best_epoch = 0
        if max_epoch:
            self.epoch = max_epoch
        for epoch in range(self.epoch):
            t1 = time()
            self.shuffle_in_unison_scary(ui_train, uv_train, ci_train, cv_train,
            total_batch = int(len(y_train) / self.batch_size)
            for i in range(total_batch):
                ui_batch, uv_batch, ci_batch, cv_batch, ai_batch, av_batch, y_bat
                self.fit_on_batch(ui_batch, uv_batch, ci_batch, cv_batch, ai_batc

            # evaluate training and validation datasets
            train_result, train_deep, train_f1 = self.evaluate(ui_train, uv_trair
            self.train_result.append(train_result)
            if has_valid:
                valid_result, valid_deep, valid_f1= self.evaluate(ui_valid, uv_va
                self.valid_result.append(valid_result)
            if self.verbose > 0 and epoch % self.verbose == 0:
                if has_valid:
```

```python
        weights["u_feat_embeddings"] = tf.Variable(
            tf.random_normal([self.u_feat_size, self.embedding_size], 0.0, 0.1),
            name="u_feature_embeddings")  # feature_size * K

        weights["c_feat_embeddings"] = tf.Variable(
            tf.random_normal([self.c_feat_size, self.embedding_size], 0.0, 0.1),
            name="c_feature_embeddings")  # feature_size * K

        #u_pool_w, u_pool_b = self._init_layer_weight((self.u_field_size+self.c_

        #weights['ctx_pool_weight'] = tf.Variable(u_pool_w, name='ctx_pool_weight

        #weights["ctx_pool_bias"] = tf.Variable(u_pool_b, name='ctx_pool_bias',dt
        #####separate user
        u_pool_w, u_pool_b = self._init_layer_weight((self.u_field_size)*self.emb
        weights['ctx_pool_weight'] = tf.Variable(u_pool_w, name='ctx_pool_weight'
        weights["ctx_pool_bias"] = tf.Variable(u_pool_b, name='ctx_pool_bias',dty


        ######create separate course pool as well
        a_conv_w, a_conv_b = self._init_layer_weight(5*self.embedding_size, self.
        weights['a_conv_filter'] = tf.Variable(np.reshape(a_conv_w, [5, self.embe
        weights['a_conv_bias'] = tf.Variable(np.reshape(a_conv_b, [1,1,self.conv_
        attn_out_1, attn_bias_1 = self._init_layer_weight(self.conv_size+self.con
        weights['attn_out_1'] = tf.Variable(attn_out_1, name='attn_out_1', dtype=
        weights['attn_bias_1'] = tf.Variable(attn_bias_1, name='attn_bias_1', dty
        attn_out, attn_bias = self._init_layer_weight(self.attn_size, 1)
        weights['attn_out'] = tf.Variable(attn_out, name='attn_out', dtype='float

        num_layer = len(self.deep_layers)
        input_size = self.conv_size + self.context_size

        glorot = np.sqrt(2.0 / (input_size + self.deep_layers[0]))
        weights["layer_0"] = tf.Variable(
            np.random.normal(loc=0, scale=glorot, size=(input_size, self.deep_lay
        weights["bias_0"] = tf.Variable(np.random.normal(loc=0, scale=glorot, si
        for i in range(1, num_layer):
            glorot = np.sqrt(2.0 / (self.deep_layers[i-1] + self.deep_layers[i])
            weights["layer_%d" % i] = tf.Variable(
                np.random.normal(loc=0, scale=glorot, size=(self.deep_layers[i-1]
                dtype=np.float32)  # layers[i-1] * layers[i]
            weights["bias_%d" % i] = tf.Variable(
                np.random.normal(loc=0, scale=glorot, size=(1, self.deep_layers[
                dtype=np.float32)  # 1 * layer[i]

        input_size = self.deep_layers[-1]
        glorot = np.sqrt(2.0 / (input_size + 1))
        weights["logistic_weight"] = tf.Variable(
                        np.random.normal(loc=0, scale=glorot, size=(input_size,
                        dtype=np.float32)  # layers[i-1]*layers[i]
        weights["logistic_bias"] = tf.Variable(tf.constant(0.01), dtype=np.float
        return weights

    def get_batch(self, ui, uv, ci, cv, ai, av, y, batch_size, index):
        start = index * batch_size
```



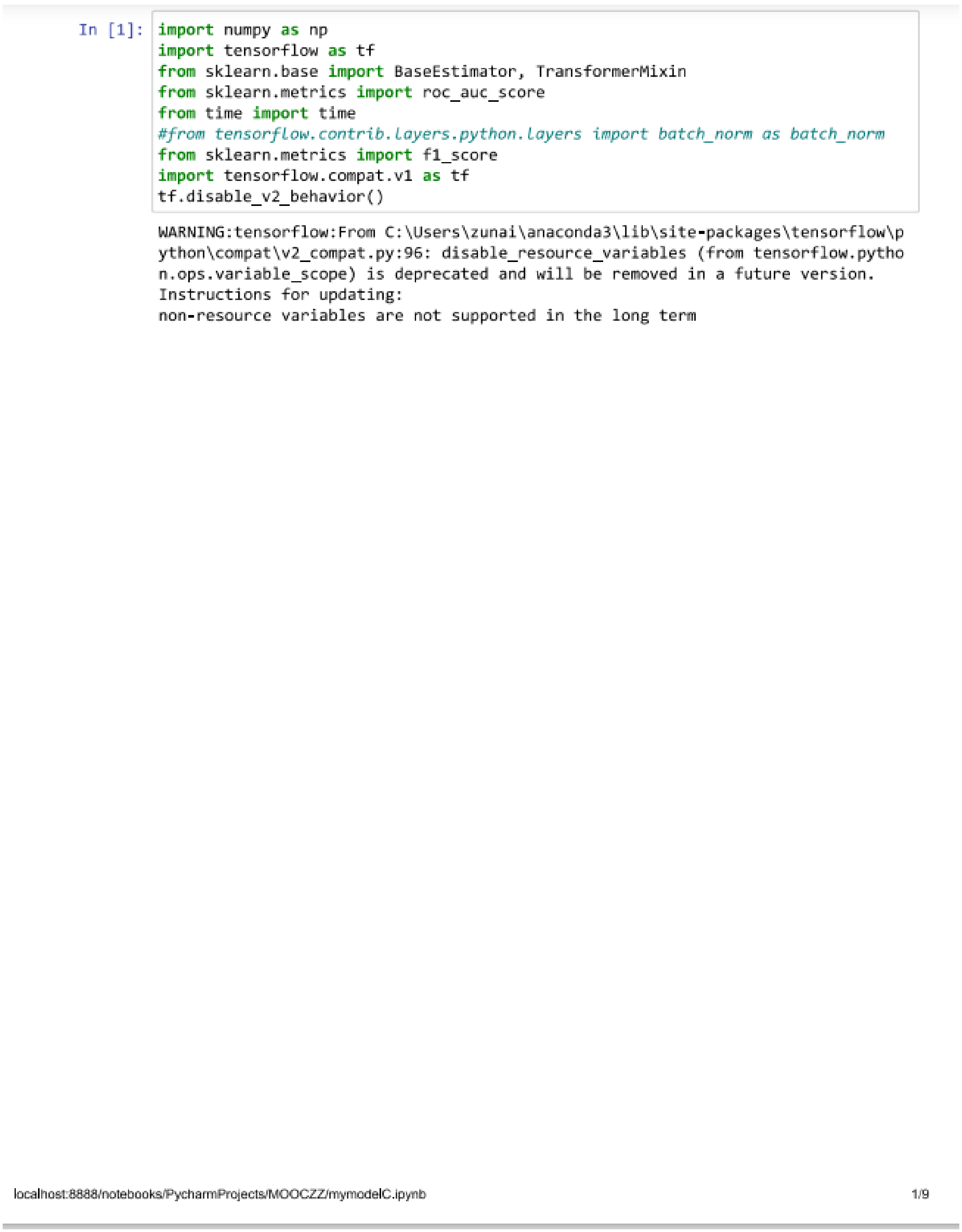
6/22/2021 mymodelC - Jupyter Notebook

```
            deep_input = tf.reduce_sum(self.a_weight_emb, axis=1)
            deep_input = tf.concat([deep_input, self.uc_inter], axis=1)

            #self.y_deep = tf.reshape(deep_input, shape=[-1,  self.conv_size])
            self.y_deep = deep_input
            self.y_deep = tf.nn.dropout(self.y_deep, self.dropout_keep_deep[0])
            for i in range(0, len(self.deep_layers)):
                self.y_deep = tf.add(tf.matmul(self.y_deep, self.weights["layer_
            #    if self.batch_norm:
             #       self.y_deep = self.batch_norm_layer(self.y_deep, train_phase
                self.y_deep = self.activation(self.y_deep)
                self.y_deep = tf.nn.dropout(self.y_deep, self.dropout_keep_deep[

            self.out = tf.add(tf.matmul(self.y_deep, self.weights["logistic_weig

            # loss
            self.out = tf.nn.sigmoid(self.out)
            self.loss = tf.losses.log_loss(self.label, self.out)
            # l2 regularization on weights
            # optimizer
            self.optimizer = tf.train.AdamOptimizer(learning_rate=self.learning_
                                                    epsilon=1e-8).minimize(self.
            self.saver = tf.train.Saver()
            init = tf.global_variables_initializer()
            self.sess = self._init_session()
            self.sess.run(init)

            total_parameters = 0
            for variable in self.weights.values():
                shape = variable.get_shape()
                variable_parameters = 1
                for dim in shape:
                    variable_parameters *= dim.value
                total_parameters += variable_parameters
            if self.verbose > 0:
                print("#params: %d" % total_parameters)


    def _init_session(self):
        config = tf.ConfigProto()
        config.allow_soft_placement = True
        config.gpu_options.allow_growth = True
        config.gpu_options.per_process_gpu_memory_fraction = 0.9
        return tf.Session(config=config)

    def _init_layer_weight(self, input_size, output_size):
        glorot = np.sqrt(2.0 / (input_size + output_size))

        return np.random.normal(loc=0, scale=glorot, size=(input_size, output_si

    def _initialize_weights(self, c_node_embedding=None, u_node_embedding=None):
        weights = dict()

        # embeddings
        weights["a_feat_embeddings"] = tf.Variable(
            tf.random_normal([self.a_feat_size, self.embedding_size], 0.0, 0.1),
```

localhost:8888/notebooks/PycharmProjects/MOOCZZ/mymodelC.ipynb 4/9

```
self.u_feat_index = tf.placeholder(tf.int32, shape=[None, None],
                                     name="u_feat_index")  # None * F
self.c_feat_index = tf.placeholder(tf.int32, shape=[None, None], name
self.a_feat_index = tf.placeholder(tf.int32, shape=[None, None], name
self.dropout_keep_attn = tf.placeholder(tf.float32, shape=[None], nam

self.u_feat_value = tf.placeholder(tf.float32, shape=[None, None], na
self.c_feat_value = tf.placeholder(tf.float32, shape=[None, None], na
self.a_feat_value = tf.placeholder(tf.float32, shape=[None, None], na

self.label = tf.placeholder(tf.float32, shape=[None, 1], name="label"
self.dropout_keep_deep = tf.placeholder(tf.float32, shape=[None], nam
self.train_phase = tf.placeholder(tf.bool, name="train_phase")

self.weights = self._initialize_weights()

# model
self.a_embeddings = tf.nn.embedding_lookup(self.weights["a_feat_embed
self.u_embeddings = tf.nn.embedding_lookup(self.weights['u_feat_embed
self.c_embeddings = tf.nn.embedding_lookup(self.weights['c_feat_embed

u_feat_value = tf.reshape(self.u_feat_value, shape=[-1, self.u_field_
c_feat_value = tf.reshape(self.c_feat_value, shape=[-1, self.c_field_
a_feat_value = tf.reshape(self.a_feat_value, shape=[-1, self.a_field_
self.c_embeddings = tf.multiply(self.c_embeddings, c_feat_value)
self.u_embeddings = tf.multiply(self.u_embeddings, u_feat_value)
self.a_embeddings = tf.multiply(self.a_embeddings, a_feat_value)
#if self.batch_norm:
#    self.a_embeddings = self.batch_norm_layer(self.a_embeddings, tra
self.a_embeddings = tf.nn.conv1d(self.a_embeddings, self.weights['a_c
                                 data_format='NWC') + self.weights['a

self.a_embeddings = tf.nn.relu(self.a_embeddings)

##Change uc to separate user and course
#self.uc_embeddings = tf.concat([self.u_embeddings, self.c_embeddings

#self.uc_inter = tf.nn.relu(tf.matmul(tf.reshape(self.uc_embeddings,
                              #      (self.u_field_size+self.c_field_
                               #      self.weights['ctx_pool_weight']

self.uc_inter = tf.nn.relu(tf.matmul(tf.reshape(self.u_embeddings, sh
                                    (self.u_field_size)*self.embeddin
                                     self.weights['ctx_pool_weight'])

self.uca_inter = tf.concat([tf.tile(tf.expand_dims(self.uc_inter, 1),
                            self.a_embeddings], 2)

self.attn_logit = tf.nn.relu(tf.matmul(tf.reshape(self.uca_inter,
                                                  shape=[-1, self.con
                                       self.weights['attn_out_1']) +
self.attn_w = tf.nn.softmax(tf.reshape(tf.matmul(self.attn_logit, sel
                                       shape=[-1, self.a_field_size/
if self.attn_enable:
    self.a_weight_emb = tf.multiply(tf.expand_dims(self.attn_w,2), se
else:
    self.a_weight_emb = self.a_embeddings
```

In [2]:
```
class CFIN():
    def __init__(self, a_feat_size, u_feat_size, c_feat_size, a_field_size, u_fie
                 embedding_size=8,
                 conv_size = 32,
                 context_size = 16,
                 deep_layers=[32, 32], dropout_deep=[0.5, 0.5, 0.5], dropout_attr
                 activation=tf.nn.relu,
                 attn_size = 16,
                 epoch=10, batch_size=256,

                 learning_rate=0.001, optimizer_type="adam",
                 batch_norm=0, batch_norm_decay=0.995,
                 verbose=False, random_seed=2016,
                 loss_type="logloss", eval_metric=roc_auc_score,
                 l2_reg=0.0, attn=True):

        self.a_feat_size = a_feat_size
        self.u_feat_size = u_feat_size
        self.c_feat_size = c_feat_size

        self.a_field_size = a_field_size
        self.u_field_size = u_field_size
        self.c_field_size = c_field_size
        self.conv_size = conv_size
        self.context_size = context_size
        self.embedding_size = embedding_size

        self.deep_layers = deep_layers
        self.dropout_deep = dropout_deep
        self.dropout_attn = dropout_attn
        self.activation = activation
        self.l2_reg = l2_reg
        self.attn_enable = attn
        self.epoch = epoch
        self.batch_size = batch_size
        self.learning_rate = learning_rate
        self.optimizer_type = optimizer_type
        self.attn_size = attn_size
        self.batch_norm = batch_norm
        self.batch_norm_decay = batch_norm_decay
        self.greater_is_better = True
        self.verbose = verbose
        self.random_seed = random_seed
        self.loss_type = loss_type
        self.eval_metric = eval_metric
        self.train_result, self.valid_result = [], []
        self._init_graph()


    def _init_graph(self):

        self.graph = tf.Graph()
        with self.graph.as_default():
            tf.set_random_seed(self.random_seed)
            #tf.random.set_seed(self.random_seed)
```



```
In [1]: import numpy as np
        import tensorflow as tf
        from sklearn.base import BaseEstimator, TransformerMixin
        from sklearn.metrics import roc_auc_score
        from time import time
        #from tensorflow.contrib.layers.python.layers import batch_norm as batch_norm
        from sklearn.metrics import f1_score
        import tensorflow.compat.v1 as tf
        tf.disable_v2_behavior()
```

```
WARNING:tensorflow:From C:\Users\zunai\anaconda3\lib\site-packages\tensorflow\p
ython\compat\v2_compat.py:96: disable_resource_variables (from tensorflow.pytho
n.ops.variable_scope) is deprecated and will be removed in a future version.
Instructions for updating:
non-resource variables are not supported in the long term
```

localhost:8888/notebooks/PycharmProjects/MOOCZZ/mymodelC.ipynb 1/9

```
            feed_dict = {self.u_feat_index: ui_batch,
                         self.u_feat_value: uv_batch,
                         self.c_feat_index: ci_batch,
                         self.c_feat_value: cv_batch,
```

```
                         self.a_feat_index: ai_batch,
                         self.a_feat_value: av_batch,
                         self.dropout_keep_deep: [1.0] * len(self.dropout_deep),
                         self.dropout_keep_attn: [1.0] * len(self.dropout_attn),
                         self.label: y_batch,
                         self.train_phase: False}
            y_deep,batch_out = self.sess.run([self.y_deep,self.out], feed_dict=fe
            if batch_index == 0:
                y_pred = np.reshape(batch_out, (num_batch,))
            else:
                y_pred = np.concatenate((y_pred, np.reshape(batch_out, (num_batch

            batch_index += 1

            ui_batch, uv_batch, ci_batch, cv_batch, ai_batch, av_batch ,y_batch =
        return y_deep,y_pred


    def evaluate(self, ui, uv, ci, cv, ai, av, y):
        y_deep, y_pred = self.predict(ui, uv, ci, cv, ai, av)
        return self.eval_metric(y, y_pred), y_deep, f1_score(y, [1 if x>0.5 else
```

In [ ]:

In [ ]: