2022

Homework 3

FINANTIAL TECHNOLOGY

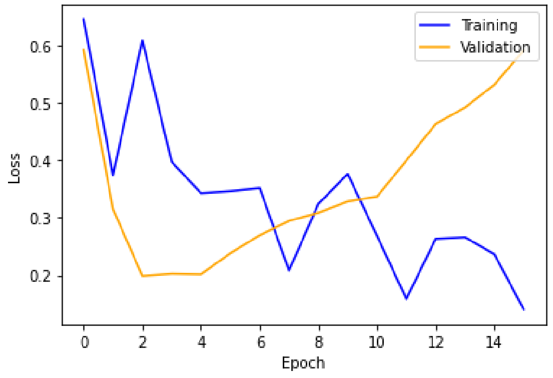
MIGUEL BENALCÁZAR – D11949001

iii. Predict the probability of clicking the push notification

iv. You have to plot the receiver operating characteristic curve (ROC), and precision-recall curve (PRC) with their area-under-curve (AUROC and AUROC), as shown in Figure 2 on the training and validation set. And show the AUROC score and AUROC score of your training, validation, and test set. Which metric do you think is more suitable for this task? Please explain why.

Bi-LSTM TRAINING RESULTS

Bi-LSTM (Bidirectional LSTM) - window = 10



Number of samples in the batch = 64
Hidden size = 30
Learning rate = 3.3e-4
Epochs = 4
Dropout = 0.5

Figure 1. Bi-LSTM Window = 10 loss curves

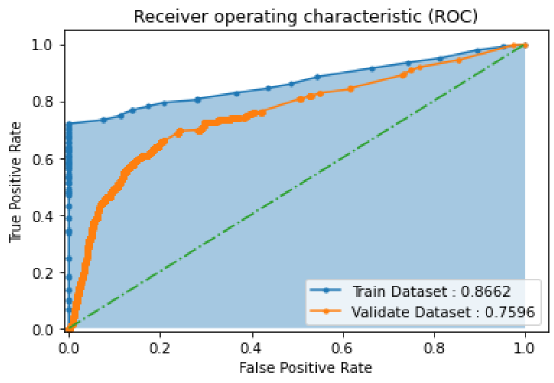


Figure 2. Bi-LSTM window = 10 ROC curves

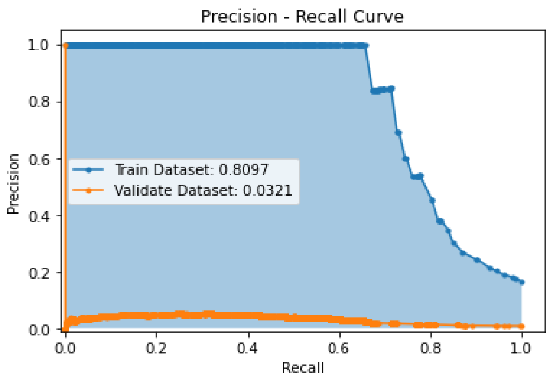


Figure 3. Bi-LSTM window = 10 Precision-recall curves

The dataset provided was highly imbalanced, the imbalance ratio was 1/100 as figure 6. shows. Thus, the current dataset needs to be shrunken. Otherwise, the training will always be biased towards the majority class. The chosen ratio for this experiment was set 1/5, which means around 5585 samples for the majority class and 1117 samples for the minority class as Figure 7 depicts.

The accuracy is not a suitable metric to evaluate this model since it is obtained dividing the total number of correct predictions by the total number of samples in the prediction. If we consider that the training dataset is still imbalanced the accuracy will be relatively high due to the model learning to predict the majority class 0 easily. Thus, this metric is suitable for balanced datasets.

The goal of this experiment is to predict the probability that a user will click after receiving each push notification. Therefore, it is crucial to increase the chances of clicking through the pushed content. Precision is a metric that represents the rate of how many are truly positives are there from the all predicted positives. Recall represents the true positive rate, in other words, it represents the ability of the model to find all TP. $F_\beta - score$ combines the Recall and Precision in one metric by taking their harmonic mean. In this way, the two metrics are compared to each other. Therefore, $F_B - score$ is the most suitable metric for imbalance datasets.

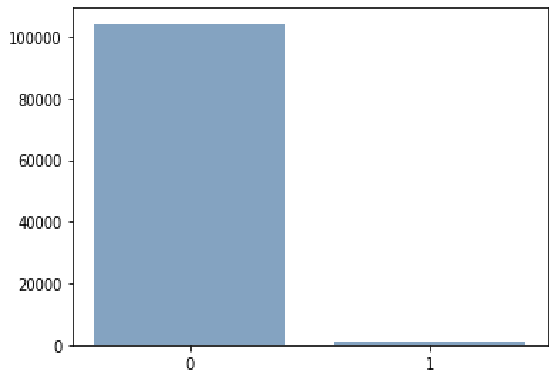


Figure 4. Original Dataset

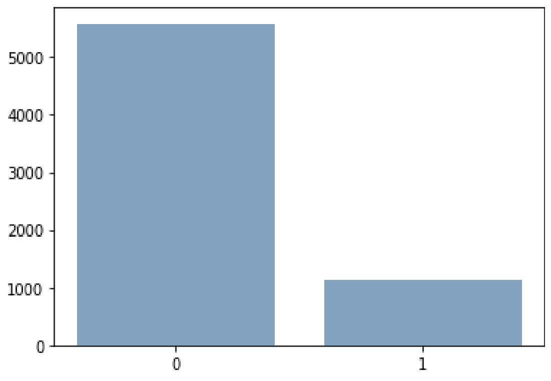


Figure 5. Under sampled Dataset

In general, the classification is simply set to 0.5, which is unsuitable for imbalanced classification. To determine the ideal threshold F_β was used. Thus, the optimized F_1 score from the Bidirectional LSTM was obtained from the testing process. Results of the testing process after using the optimal threshold are shown in Table. 1.

As it can be seen in Table. 1, f1-score is low due to the precision obtained. The precision depends on the total of TP divided by the total of (TP + FP). If the total of FP is relatively high, the precision will be low. This effect occurs because of the dataset nature. If we want a better performance from the model, it needs to classify better the TN and FP. In our experiment, when TN is well classified, FP is low, and thus TP is low as well.

For this experiment, TPR or recall must be high because we want to maximize the clicked rate. The model, however, performs well in terms of recall but poorly in terms of precision, yielding a low F1-score. As a result, the model was trained with a focus on recall while also attempting to optimize the precision score. Hence, the precision-recall curve and its area under the curve is more important than AUROC.

Bi-LSTM MODEL

To implement this model a single bidirectional LSTM layer is defined, then the output of the Bi-LSMT layer feeds a linear layer and as last layer the sigmoid.

```
class LSTM_Model(nn.Module):
    def __init__(self, input_size, hidden_size = 2, num_layers = 1, dropout = 0, output_size = 1, device = 'gpu'):
        super(LSTM_Model, self).__init__()

        self.device = getDevice(device)
        self.hidden_size = hidden_size
        self.num_layer = num_layers
        self.lstm = nn.LSTM(
            input_size = input_size,
            hidden_size = hidden_size,
            num_layers = num_layers,
            batch_first = True,
            dropout =  dropout,
            bidirectional = True
        )
        self.fc = nn.Linear(hidden_size * 2, output_size)
        self.out = nn.Sigmoid()
        nn.init.xavier_uniform_(self.fc.weight)

    def forward(self, x):
```

```
h0 = Variable(torch.zeros(2 * self.num_layer, x.size(0), self.hidden_size)).to(self.device)
c0 = Variable(torch.zeros(2 * self.num_layer, x.size(0), self.hidden_size)).to(self.device)
out , (hn, cn) = self.lstm(x, (h0,c0))
out = self.fc(out[:, -1, :])
out = self.out(out)
return out
```

Bi-LSTM TESTING RESULTS

Bi-LSTM (Bidirectional LSTM) - Testing Model Results - Threshold = 0.399

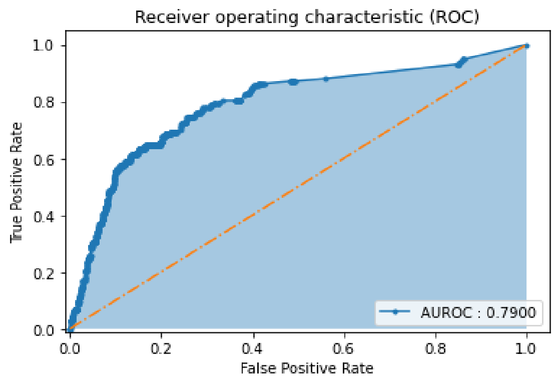


Figure 6Bi-LSTM window = 10 ROC curves Testing

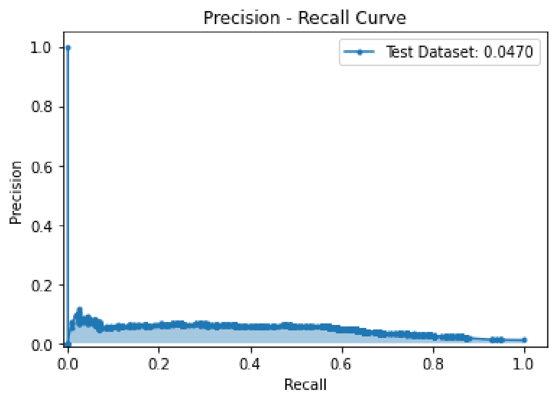


Figure 7 . Bi-LSTM window = 10 Precision-recall curves testing

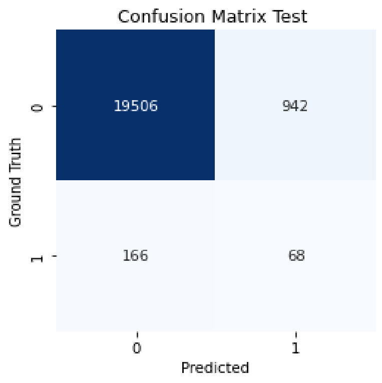
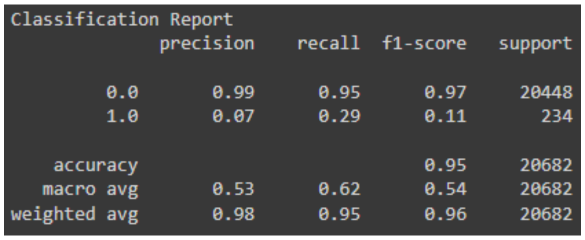


Figure 8. Bi-LSTM window = 10 Testing Confusion Matrix



```
Classification Report
                  precision    recall  f1-score   support

         0.0          0.99      0.95      0.97     20448
         1.0          0.07      0.29      0.11       234

    accuracy                              0.95     20682
   macro avg          0.53      0.62      0.54     20682
weighted avg          0.98      0.95      0.96     20682
```

Figure 9. Bi-LSTM window = 10 classification report for testing

Table 1. Bi-LSTM, threshold = 0.399

Precision	*Recall*	*F1-score*	*AUROC*	*PRC*
0.0256	0.07	0.11	0.79	0.047

v. **Please replace the bidirectional-LSTM cell in your click prediction model with a bidirectional-GRU and repeat (iii), (iv).**

Bi-GRU MODEL

Like the Bi-LSTM model, this model has a single bidirectional GRU layer to implement it. Then the output of the Bi-GRU layer feeds a linear layer and as the last layer, the sigmoid.

```
class GRU_Model(nn.Module):
  def __init__(self, input_size, hidden_size = 2, num_layers = 1, dropout = 0, output_size = 1, device = 'gpu'):
    super(GRU_Model, self).__init__()

    self.device = getDevice(device)
    self.hidden_size = hidden_size
    self.num_layer = num_layers

    self.gru = nn.GRU(
      input_size = input_size,
      hidden_size = hidden_size,
      num_layers = num_layers,
      batch_first = True,
      dropout =  dropout,
      bidirectional = True
    )

    self.fc = nn.Linear(hidden_size * 2, output_size)
    self.out = nn.Sigmoid()
    nn.init.xavier_normal_(self.fc.weight)

  def forward(self, x):

    h0 = Variable(torch.zeros(2 * self.num_layer, x.size(0), self.hidden_size)).to(self.device)

    out, hn = self.gru(x, h0)
    out = self.fc(out[:, -1, :])
    out = self.out(out)
    return out
```

Bi-GRU TRAINING RESULTS

Bi-GRU (Bidirectional GRU) - window = 10 Results

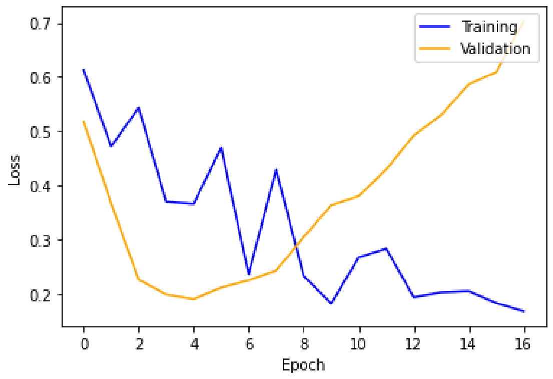


Figure 10. Bi-GRU Window = 10 loss curves

Number of samples in the batch = 64
Hidden size = 30
Learning rate = 1.3e-4
Epochs = 5
Dropout = 0.5

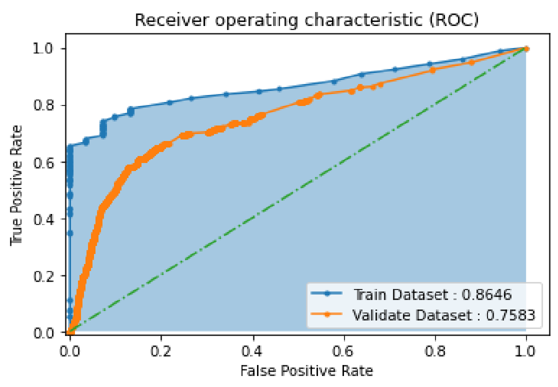


Figure 11. Bi-GRU window = 10 ROC curves

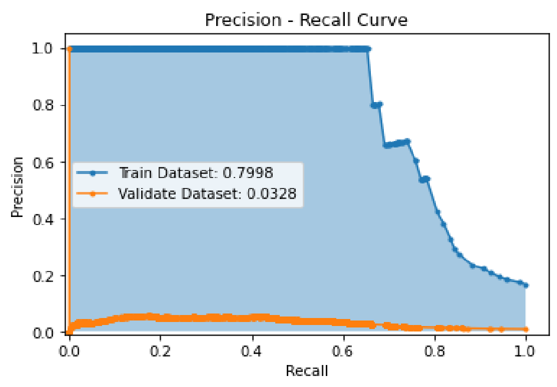


Figure 12. Bi-GRU window = 10 Precision-recall curves

For this case, the threshold is 0.411. Results of the testing process are shown in Table. 2.

Bi-GRU TESTING RESULTS

Table 2. Bi-GRU, threshold = 0.411

Precision	*Recall*	*F1-score*	*AUROC*	*PRC*
0.06	0.41	0.11	0.7917	0.0444

Bi-GRU (Bidirectional GRU) - Testing Model Results - threshold = 0.411

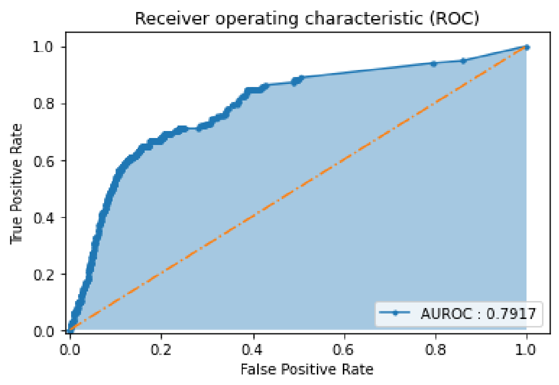


Figure 13. Bi-GRU window = 10 ROC curves Testing

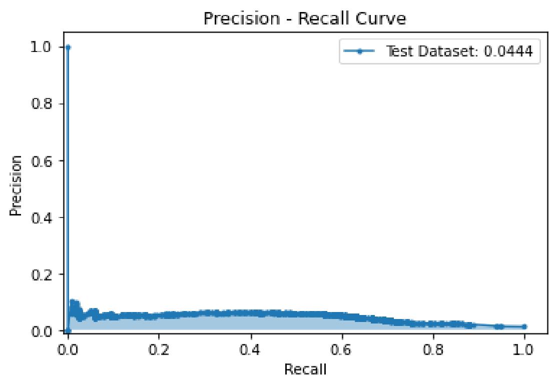


Figure 14. Bi-GRU window = 10 Precision-recall curves testing

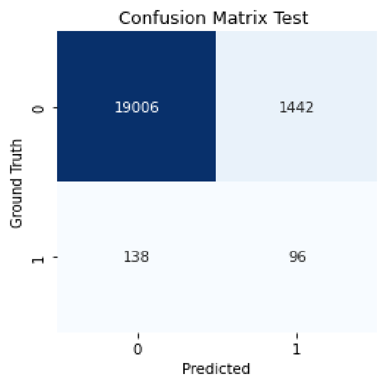
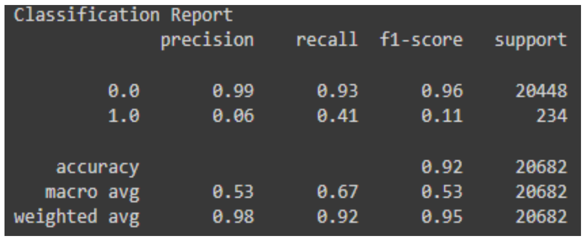


Figure 15. Bi-GRU window = 10 Testing Confusion Matrix



```
Classification Report
              precision    recall  f1-score   support

         0.0       0.99      0.93      0.96     20448
         1.0       0.06      0.41      0.11       234

    accuracy                           0.92     20682
   macro avg       0.53      0.67      0.53     20682
weighted avg       0.98      0.92      0.95     20682
```

Figure 16. Bi-GRU window = 10 classification report for testing

vi. **(15%) Please replace the bidirectional-GRU cell in your click prediction model with a transformer encoder block and repeat (iii), (iv).**

TRANSFORMER MODEL

This model uses three transformer encoder blocks using 'gelu' as the activation function of each transformer encoder, they share the same number of heads and 0,2 as dropout for the first two encoders while 0.5 to the last layer. The results of the last layer are fed to a linear layer and a sigmoid.

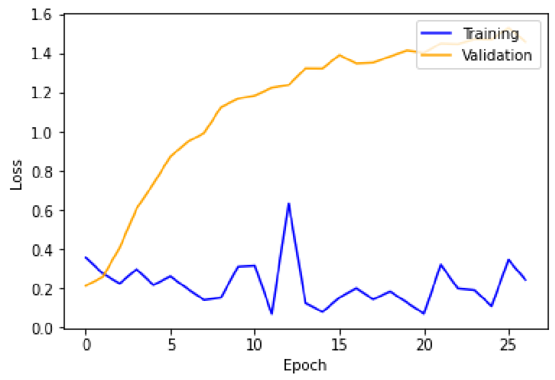
```
class Transformer_Model(nn.Module):
    def __init__(self, d_model, nhead, device = __TEST_DEVICE__):

        super(Transformer_Model, self).__init__()

        self.device = getDevice(device)
        self.encoder = nn.TransformerEncoderLayer(
            d_model  = d_model,
            nhead = nhead,
            dropout = 0.2,
            activation = 'gelu',
            batch_first = True
        )

        self.encoder1 = nn.TransformerEncoderLayer(
            d_model  = d_model,
            nhead = nhead,
            dropout = 0.2,
            activation = 'gelu',
            batch_first = True
        )

        self.encoder2 = nn.TransformerEncoderLayer(
            d_model  = d_model,
            nhead = nhead,
            dropout = 0.5,
            activation = 'gelu',
            batch_first = True
        )

        self.fc = nn.Linear(d_model, 1)
        self.out = nn.Sigmoid()

        nn.init.xavier_normal_(self.fc.weight)

    def forward(self, x):
        out = self.encoder(x)
        out = self.encoder1(x)
        out = self.encoder2(x)
        out = self.fc(out[:, -1, :])
        out = self.out(out)
        return out
```

TRANFORMER TRAINING RESULTS

Table 3. Transformer, threshold = 0.997

Precision	*Recall*	*F1-score*	*AUROC*	*PRC*
0.05	0.6	0.09	0.7797	0.0430

Transformer encoder block - window = 10 Results



Number of samples in the batch = 64
Learning rate = 1.2e-4
Number of heads = 9
Epochs = 15

Figure 17. Transformer Window = 10 loss curves

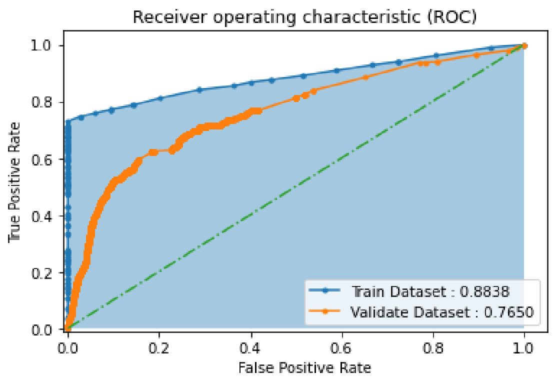


Figure 18. Transformer window = 10 ROC curves

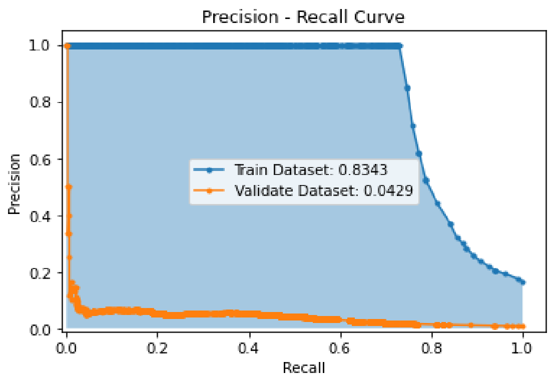


Figure 19. Transformer window = 10 Precision-recall curves

TRANSFORMER TESTING RESULTS

Transformer encoder block Testing Model Results - threshold = 0.997

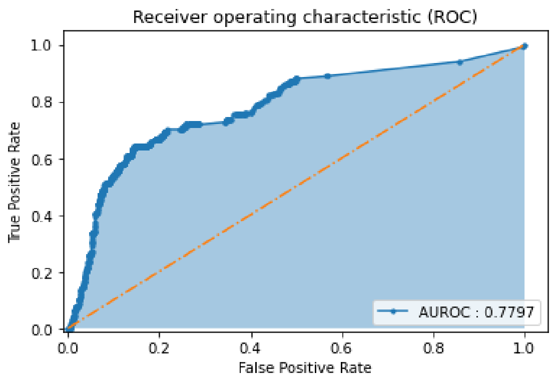


Figure 20.Transformer window = 10 ROC curves Testing

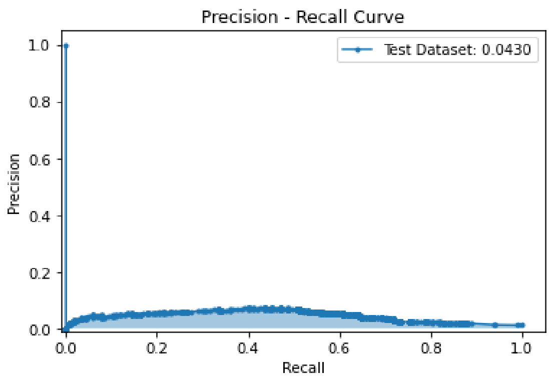


Figure 21. Transformer window = 10 Precision-recall curves testing

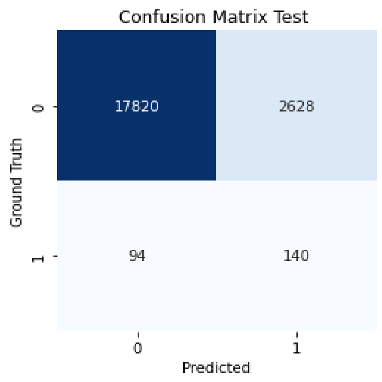
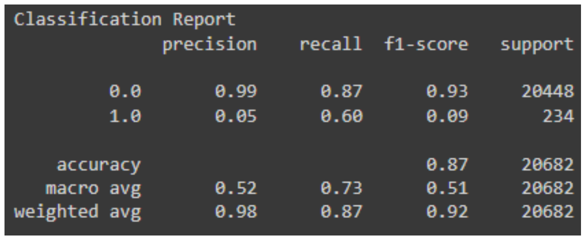


Figure 22. Transformer window = 10 Testing Confusion Matrix



```
Classification Report
              precision    recall  f1-score   support

         0.0       0.99      0.87      0.93     20448
         1.0       0.05      0.60      0.09       234

    accuracy                           0.87     20682
   macro avg       0.52      0.73      0.51     20682
weighted avg       0.98      0.87      0.92     20682
```

Figure 23. Transformer window = 10 classification report for testing

vii. Please use the metric you choose in (iv) to compare these three models (Bidirectional-LSTM, Bidirectional-GRU, Transformer) and briefly describe what you found.

Table 4. Results of the model

Models	*Precision*	*Recall*	*F1-score*	*AUROC*	*PRC*
Bi-LSTM	0.07	0.29	0.11	0.79	0.047
Bi-GRU	0.06	0.41	0.10	0.7917	0.0444
Transformer	0.05	0.6	0.09	0.7797	0.0430

Table 4. displays the outcomes of various metrics from the three different models. In this case, we must rely on the f1-score, but this is directly related to the precision and recall results. As a result, the table includes all these metrics. As can be seen, Bi-LSTM reached the highest F1-score but the lowest recall. Hence, it can be concluded based on the results that transformer model has better results in the recall and it also shows pretty good results in precision, so F1-score is acceptable base on the comparisons with other models tested.

viii. In the beginning, we mentioned in the impression dataset, 'click_seq_10' means the last 10 click records from now on, and 'click_seq_10_onehot' is their one-hot vector. Please try to replace this feature with the last 3 and 5 click records. (You need to generate the two datasets by yourself.) Then train your model again. (You can choose one from Bidirectional-LSTM, Bidirectional-GRU, or Transformer) and compare the performance of the model

In section vii, table 4 shows the results of the transformer model which has a good performance. Therefore, the following results belongs to the Transformer by using the same hyperparameters, but with a different window size. The under-sampling ratio is also the same as the previous experiment with 1/5.

TRANSFORMER TRAINING RESULTS WINDOW = 5

Transformer - window = 5 - epoch = 16

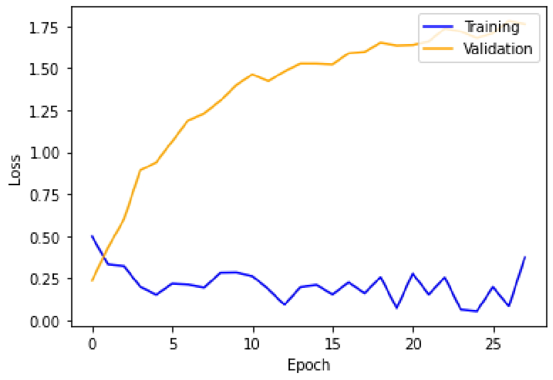


Figure 24. Transformer Window = 5 loss curves

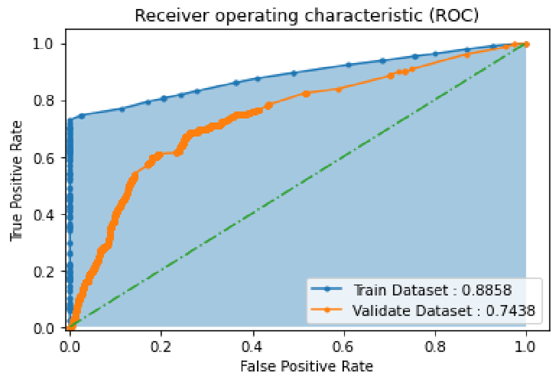


Figure 25. Transformer window = 5 ROC curves

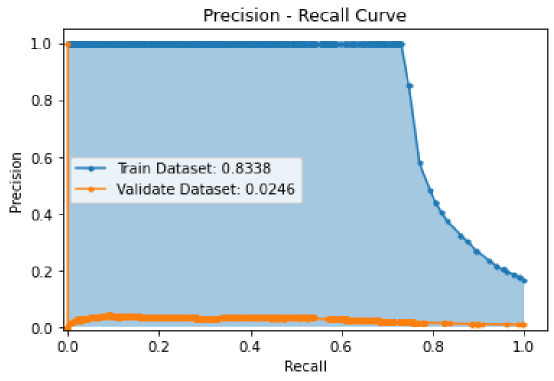


Figure 26. Transformer window = 5 Precision-recall curves

TRANSFORMER TESTING RESULTS WINDOW = 5

Table 5. Transformer window 5, threshold = 0.99

Precision	*Recall*	*F1-score*	*AUROC*	*PRC*
0.05	0.46	0.10	0.7862	0.0397

Transformer Testing Model Results - window = 5 - threshold = 0.99

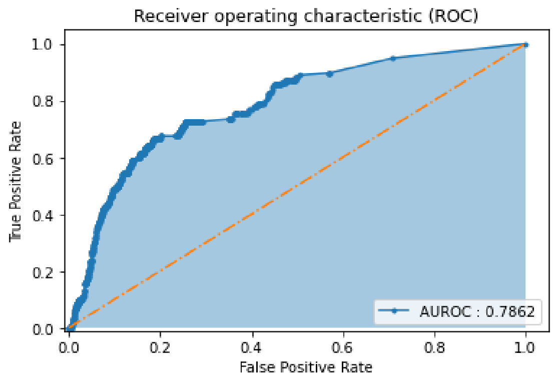


Figure 27. Transformer window = 5 ROC curves Testing

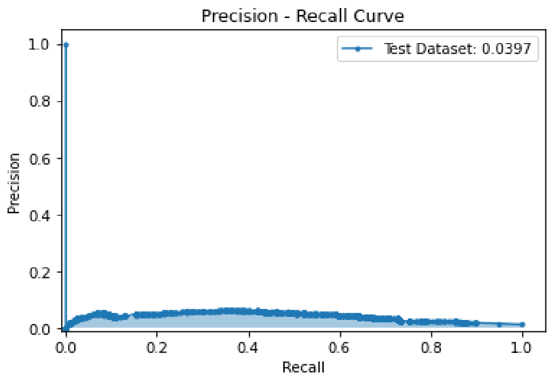


Figure 28 . Transformer window = 5 Precision-recall curves testing

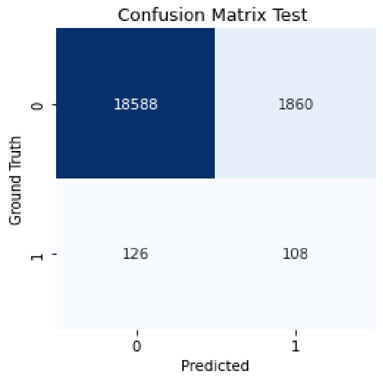
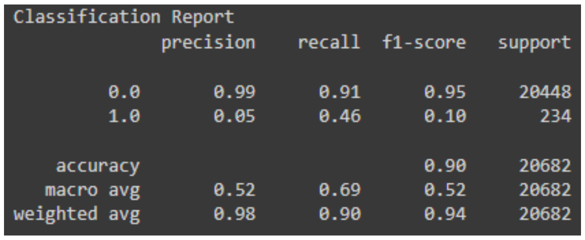


Figure 29. Transformer window = 5 Testing Confusion Matrix



```
Classification Report
                 precision    recall  f1-score   support

         0.0       0.99      0.91      0.95     20448
         1.0       0.05      0.46      0.10       234

    accuracy                           0.90     20682
   macro avg       0.52      0.69      0.52     20682
weighted avg       0.98      0.90      0.94     20682
```

Figure 30 Transformer window = 5 classification report for testing

TRANSFORMER TRAINING RESULTS WINDOW = 3

Transformer - window = 3 - epoch = 25

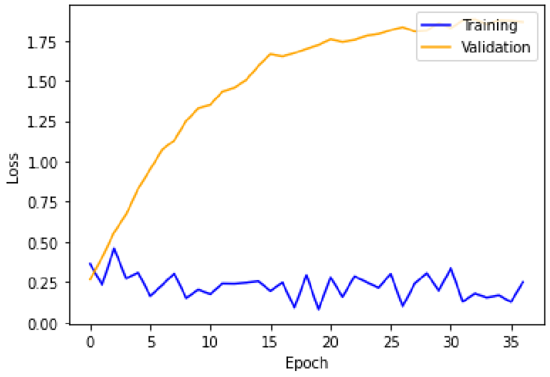


Figure 31. Transformer Window = 3 loss curves

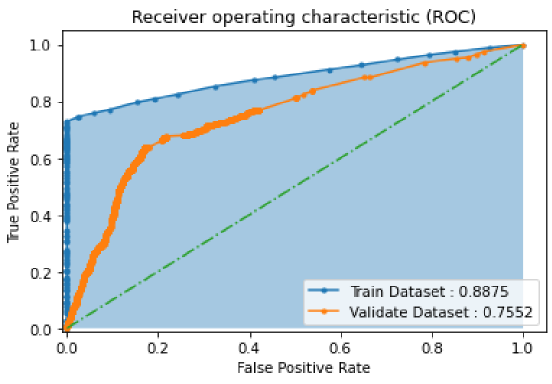


Figure 32. Transformer window = 3 ROC curves

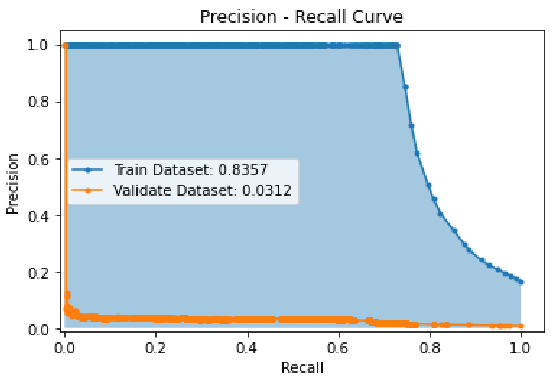


Figure 33. Transformer window = 3 Precision-recall curves

TRANSFORMER TESTING RESULTS WINDOW = 3

Table 6. Tranformer window 3, threshold = 0.998

Precision	*Recall*	*F1-score*	*AUROC*	*PRC*
0.04	0.56	0.07	0.7591	0.0279

Transformer Testing Model Results - window = 3 - threshold = 0.998

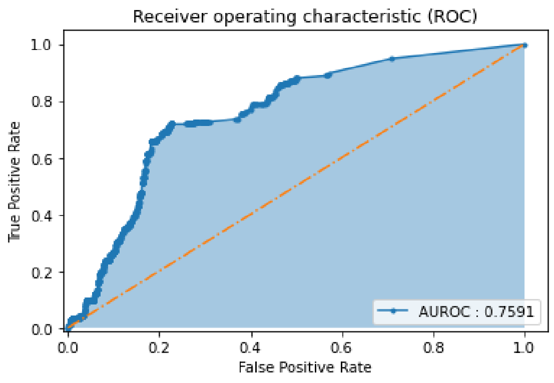


Figure 34. Transformer window = 3 ROC curves

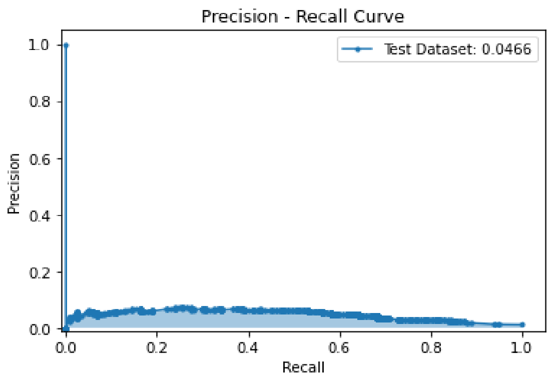


Figure 35 . Transformer window = 3 Precision-recall curves testing

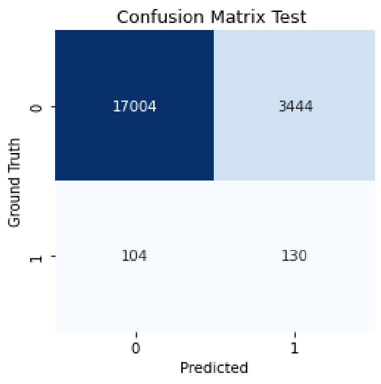
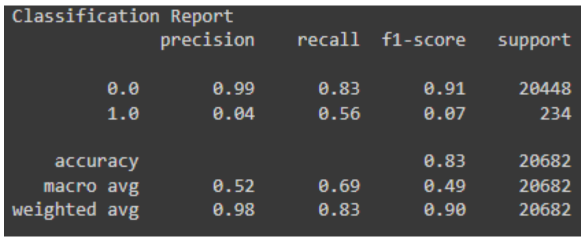


Figure 36. Transformer window = 3 Testing Confusion Matrix



```
Classification Report
              precision    recall  f1-score   support

         0.0       0.99      0.83      0.91     20448
         1.0       0.04      0.56      0.07       234

    accuracy                           0.83     20682
   macro avg       0.52      0.69      0.49     20682
weighted avg       0.98      0.83      0.90     20682
```

Figure 37. Transformer window = 3 classification report for testing

The comparison of the Transformer model with different windows size are shown in the following table.

Table 7. Transformer model performance by using different window in the dataset

Window	Precision	Recall	F1-score	AUROC	PRC
10	0.05	0.6	0.09	0.7797	0.0430
5	0.05	0.46	0.10	0.7862	0.0397
3	0.04	0.56	0.07	0.7591	0.0279

Table 7 displays the results of the Transformer model performance when a different window in the dataset is used. By simply taking the last 5 and 3 clicks from the array of 10 clicks, two new datasets were created. In this table, the Transformer model with a window of 5 performs marginally better than the results with a window of 10 and 3 base on f1-score. However, the Transformer result with a window length of 3 is better than the other results in predicting the true positive values, the recall is much better than the results obtained by the transformer model with window size of 10 and 3.

In addition to the Transformer, the other models were tested, and the results are as follows:

Table 8. Bi-LSTM model performance by using different window in the dataset

Window	Precision	Recall	F1-score	AUROC	PRC
10	0.07	0.29	0.11	0.79	0.0470
5	0.08	0.39	0.14	0.8039	0.0576
3	0.06	0.52	0.11	0.7965	0.0466

Table 9. Bi-GRU model performance by using different window in the dataset

Window	Precision	Recall	F1-score	AUROC	PRC
10	0.06	0.41	0.11	0.7917	0.0444
5	0.07	0.34	0.12	0.8006	0.0483
3	0.06	0.40	0.10	0.7901	0.0398

Table 8 and 9 shows the comparison of the model under the dataset with different window size. F1-score results to be highest when the window is 5 for all the tested models, although, the recall is not so good as the shown by the Bi-GRU with 10 as window size and Bi-LSTM with 3 as window size.

ix. **In this homework, we only use 'articleSection' and 'weekday' as news-related features. However, some information like 'keywords' and 'title' are also important features we can use. Please use your creativity to add these features to the model. (you can explain your ideas or directly implement them, the latter gets more points.)**

The impression dataset, in addition to the news list dataset, was used in this section. The click_seq, which contains the news index encode in the impression dataset, can be used to determine which news were pushed. As a result, keywords, article section, and title can be extracted and concatenated with the data from the impression dataset.

For this part a new model was created and it is display in figure 48.

Since the nature of the data is different from the array of clicks and the push notification, the model has two parts. the first one to refine and perform the feature extraction from the click sequences dataset, while the second part does the same feature extraction from the push notification that are the news (keywords, title, article section). This part must process the keywords by using BERT Chinese model to extract the embeddings and become natural information to feed the second part. The feature extraction is made by the stacking of four transformer encoder layers. At the end of each layer, the results of the four stacking layers are added into a node before feeding the linear layer. This linear layer will reduce the space from 768 which is the length to 128. Another linear layer reduces the space from 128 to 64. At the end the result of the two parts of the model are concatenated and fed to the last linear layer and the a sigmoid will return the probability.

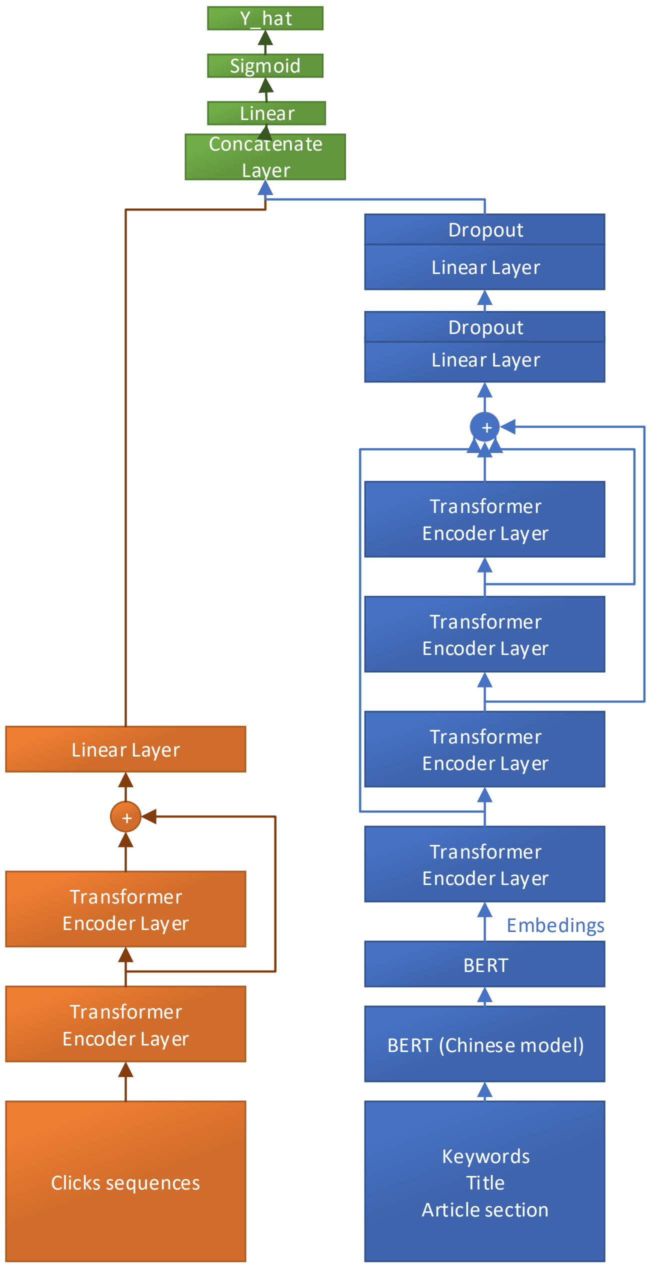
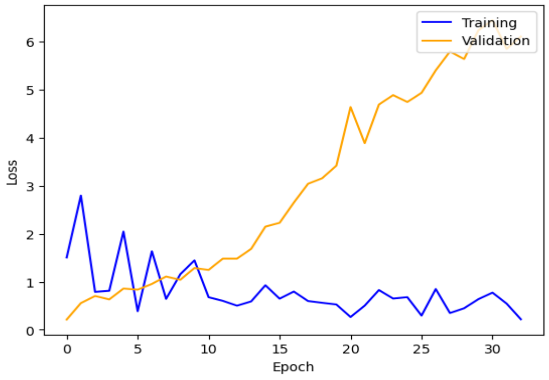


Figure 38. Bonus Model

RESULTS

Transformer encoder block, window = 3 Results



Number of samples in the batch = 512
Learning rate = 3.3e-6
Number of heads clicks part = 1
Number of heads news part = 3
Epochs = 29

Figure 39.Bonus transformer loss curves

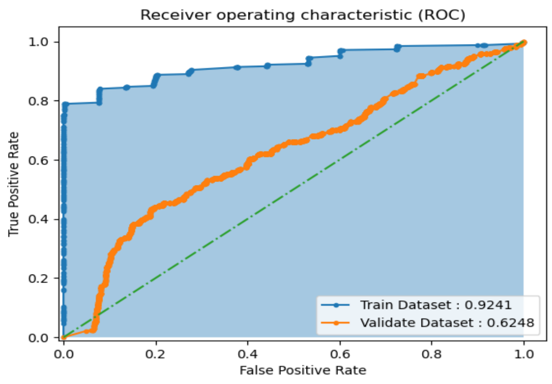


Figure 40. Bonus transformer ROC curves

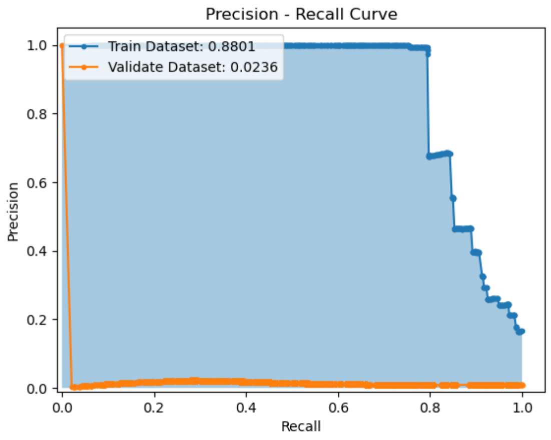


Figure 41. Bonus transformer Precision-recall curves

Table 10. Bonus transformer window = 3, g-mean = 0.5577/ f1-score = 0.4683

Window	*Threshold*	*Precision*	*Recall*	*F1-score*	*AUROC*	*PRC*
3	0.999	0.04	0.4	0.07	0.6854	0.0475
3	0.958	0.02	0.624	0.038	0.6854	0.0475

Table 10. shows results from the bonus model by using different threshold. The best threshold for this case is 0.999, with it the model can reach 0.07 f1-score which is much higher that the obtained by using 0.958 as threshold. However, the model with 0.958 as threshold reaches 0.624 from recall which is high if we compare with 0.4 obtained results by the threshold 0.999.

Transformer encoder block, Testing Model Results - window = 3 – threshold = 0.999

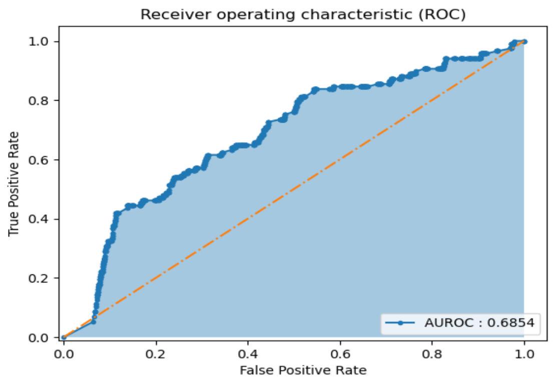


Figure 42. Bonus Transformer window = 3 ROC curves Testing

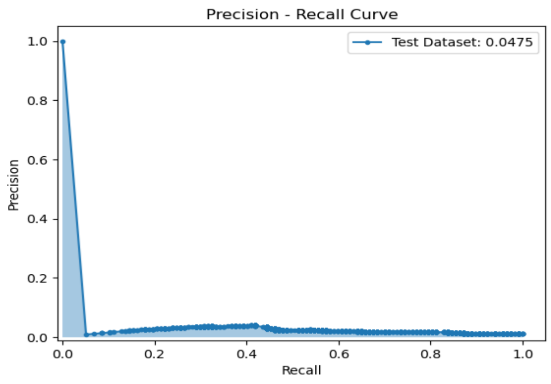


Figure 43 . Bonus Transformer window = 3 Precision-recall curves testing

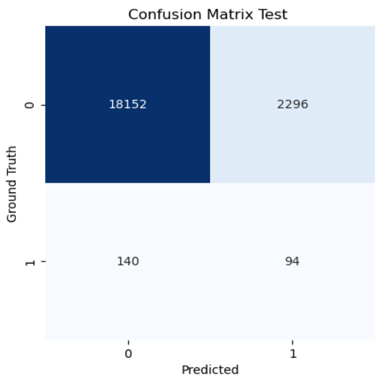


Figure 44. Bonus Transformer window = 3 Testing Confusion Matrix

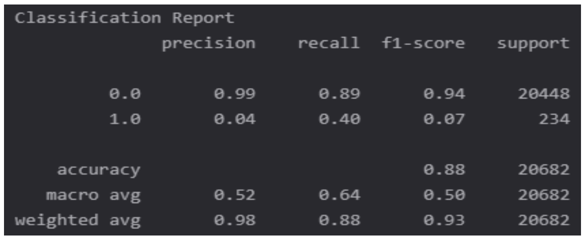

```
Classification Report
              precision    recall  f1-score   support

         0.0       0.99      0.89      0.94     20448
         1.0       0.04      0.40      0.07       234

    accuracy                           0.88     20682
   macro avg       0.52      0.64      0.50     20682
weighted avg       0.98      0.88      0.93     20682
```

Figure 45. Bonus Transformer window = 3 classification report for testing