Game Churn Prediction Project

Please read or run the ff. in order:

a. Flexible Preprocessing
b. Supervised Learning Preprocessing and prediction
c. Deep learning preprocessing and prediction

II. Project Overview

a. Definition

Churn is defined as a situation in which customers stop subscribing to a product or service of a company. In the case of this project, a churned user is a user who has stopped playing for a certain amount of time. Determining whether a customer is a churned user depends on the business case and implementation.

Approach:

1. A churned user could be determined by setting ranges. We could set an observation period and a churn period. We summarize each user's data by the observation period and use the churn period to determine whether a user is churned.
2. An alternative could be using the idea of observation period but instead of summarizing each players data, we just determine how active a user is based on time, effectively making it a time series problem.
    1. Based on the activity of every user, we can to predict the when the user will next be active next. A prediction period will have to be added to determine how far into the future we want our model to predict.
    2. We can add a churn period and determine if the user classifies as active during that period
3. A churned user could be determined by only setting a churn period. For every record of a user, we could determine how long it has been since the user last played. With this, for every user, if the time exceeds the churn period, a user can be considered churned.
4. Another approach is determining users who are truly churned based on all available data then converting our classifications to clusters. For every new user, we determine the classification based on the cluster.

b. Problem

The main problem I would like to tackle is determining whether a user is active or not. Determining how frequent a user is, is an approach that's easier to deal with once we can accurately determine if a user is active or not. I will be making use of approach 1 and 2.2 of the previous section to solve the problem.

For this project I will work on 2 case scenarios:

1. **Immediate churners**- determining churners who are not likely to play after a few days. These are users who tried the game but did not like the game.
2. **Gradual churners** - determining churners who are not likely to play after a few weeks. These are users who got bored of playing the game after some time.

c. Metrics

I will be using a few metrics while working on the solution. However, I will just only be using UAC ROC score to determine the best model.

Metrics include the following:

1. **AUC-ROC score**- AUC-ROC score is a good metric for binary classification problems because it only cares about the true classifications and it can tells us whether our model is doing better than random or not.
2. **Logarithmic loss**- Log loss is another good metric, it measures the accuracy of a classifier by penalizing false classifications. Attaining a good log loss would mean that we are generalizing all categories well and not just favoring one classification over the other.
3. **Confusion Matrix**- Confusion matrix on the other hand simply tells us where our model is classifying our labels. This metric is more helpful when building the model as its easier to see where our model is lacking.

III. Analysis

a. Data Exploration

Our raw data contains records for just about a months' time. There are 3 tables, one containing the general information, one with player details and the last one with team details. We have approximately 27127 rows in the general table, while player data has 229742 rows of which 14629 are unique players. Team data has just 54254 rows.

```
data date range 2015-05-25 17:49:57  -  2015-06-20 05:42:38
# of records in raw data 27127
# of records in player data 229742
# of unique players 14629
# of records in team data 54254
```

General Table

	date	duration	finished	group	mapId	official	players	port	server	teams	timeLimit
0	1432576197	22562	1		6	1	[{'auth': True, 'name': 'RoDyMaRy', 'flair': 1...	8008	tagpro-chord.koalabeast.com	[{'name': 'Red', 'score': 1, 'splats': 'mp59z/...	12
1	1433101992	7202	1	redacted	17	1	[{'auth': True, 'name': 'DUSTY', 'flair': 105,...	8001	tagpro-orbit.koalabeast.com	[{'name': 'Red', 'score': 1, 'splats': 'q9Mtq8...	2
2	1433885981	6100	1		24	1	[{'auth': True, 'name': 'Roll Player', 'flair'...	8008	tagpro-radius.koalabeast.com	[{'name': 'Red', 'score': 3, 'splats': 'm0Uqis...	12

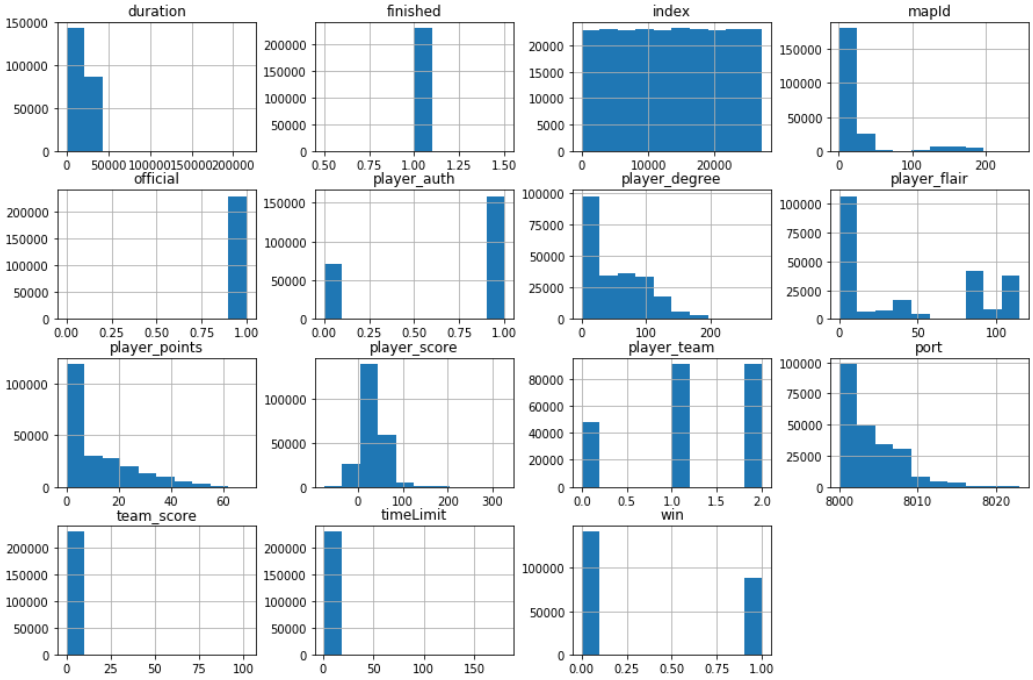
Player data

	player_auth	player_degree	player_events	player_flair	player_name	player_points	player_score	player_team	date	duration	mapId
0	True	99	AREVYIpPCAe0QCBSIAisARCvARK/ARCjARArIAgDARAfAR...	105	RoDyMaRy	25	31	1	1432576197	22562	6
1	True	0	AilApQCKggCIqwCI5wCIMwEIPABQIABRGwBUlQgE3ECgKg...	86	BartimaeusJr	0	56	2	1432576197	22562	6
2	True	31	QBCUIAk3ARI8ARCjAFErCASzQKBTARHAARGTCAdbQCDKQB...	104	2Pop	29	19	2	1432576197	22562	6

Team data

	team_name	team_score	team_splats	date	duration	mapId
0	Red	1	mp59z/DaK5eWsvU1WURi5P0ufn/XUS4TVqOBt9RPBWsOip...	1432576197	22562	6
1	Blue	3	psKLSdkU5QI2Mj1JaTciTKnUgn4nLT/TWKiVq85VLOXqar...	1432576197	22562	6
2	Red	1	q9Mtq8slpC35lbK1i/VLIHKal6Kh	1433101992	7202	17

As we can see from the table below, the distribution of several columns is pretty uneven



b. Regression vs Classification

1. **Regression**:

Pros:

1. Can naturally understand patterns in our data
2. Can identify users that are truly inactive.
3. Can yield better results over time
4. Can tell us not only if the user is churned but also how frequent a returning user is.

Cons:

1. Can take significantly longer to preprocess, tune and compute.
2. Results can be harder to interpret
3. May require more data to produce results that are better than that of a classification.

2. **Classification**:

Pros:

1. Can produce good results with little amount of data
2. Definition of churned users can be more flexible
3. Easier to process and tune
4. Easy to interpret results

Cons:

1. It may oversimplify the solution
2. Churned users may not truly be churned users

3. Time related patterns are hard to work with
4. Provides no additional value than to solve the problem

I have decided to work on a classification problem over regression. Time constraint being the first reason. I don't think I have enough time to work on a regression problem especially if I want to use time series. Working with just a month's worth of data may prove to be more challenging for regression. Based on the problem, we only need to know if a user will continue playing or not and classification can do the job just fine.

IV. Data Preprocessing

a. Feature preprocessing:

The 3 tables were merged into one table then ff was done:

1. **Features Added**
    1. **Win –**a feature created by setting a winner for the team with the highest score for every game.
2. **Features Removed**
    1. **Logical reason-** Port, map Id, server, player events, team splats, player flair, team name, player_auth, time Limit
    2. **Duplicate data-** players, teams
    3. **Sparse data-** finished
3. **Features Kept**
    1. Index, date, duration, player degree, player name, player points, player score, team score
4. **Sample output:**

	index	date	duration	player_degree	player_name	player_points	player_score	team_score	win
2	0	2015-05-25 17:49:57	22562	31	2Pop	29	19	3.0	1.0
6	0	2015-05-25 17:49:57	22562	89	Android	31	50	3.0	1.0
1	0	2015-05-25 17:49:57	22562	0	BartimaeusJr	0	56	3.0	1.0

b. Supervised Learning Classifier Preprocessing:

The data had to be furthered preprocessed, so it could work with our supervised classification models. For every player, the records were summarized based into one based on the observation period. Features added were simply based on logic. The features that were added were then filtered down to only a few features based the feature correlation plot and variable importance plot of our supervised classification models.

1. **Features Added**
    1. **Churned –** This is the feature determining whether a user is churned based on the parameters, observation period, churn period and bin time.
    2. Games played, Min score, Mean score, max score, sd score, score ratio, total time played, min game duration, avg game duration, max game duration, min player degree, avg player degree, max player degree, min player rank, avg player rank, max player rank, average distance between visit, max distance between visit, min distance between visit, std distance between visit, total games won, win ratio
2. **Features Removed**

1. Games played, min game duration, avg game duration, min player degree, avg player degree, max player degree, min player rank, avg player rank, average distance between visit, total games won, win ratio

3. **Features Kept**
    1. Mean score, games played, total time played, max player rank, max distance between visit, std distance between visit, max game duration, player
4. **Sample output:**

	mean_score	games_played	total_time_played	max_player_rank	max_dist_bw_visit	std_dist_bw_visit	max_game_duration	player	churned
0	19.000000	1	376.033333	29	0.00	0.000000	376.033333	2Pop	0
1	50.000000	1	376.033333	31	0.00	0.000000	376.033333	Android	1
2	40.727273	11	3150.600000	0	4380.65	1394.048465	720.116667	BartimaeusJr	0

c. Deep Learning Classifier Preprocessing:

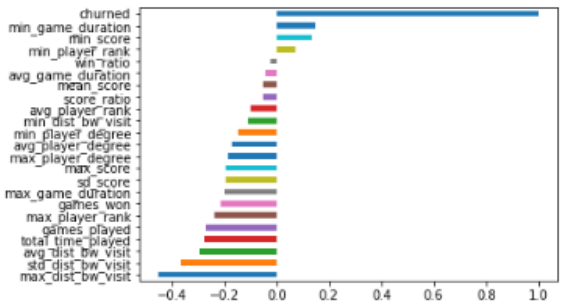
The data had to be furthered preprocessed, so it could work with our time series classification model. Every column of our record is a time frame that depends on the set bin time. The start time of every record is the first record of the player.

1. **Features Added**
    1. **Churned –** This is the feature determining whether a user is churned based on the parameters, observation period, churn period and bin time.
2. **Features Removed**
    1. Index, date, duration, player degree, player name, player score, team score
3. **Features kept**
    1. **player_points –** Since only one feature could be selected for our time series, I kept player points as it explains a little variance in our data based on the correlation plot of the classification problem.
4. **Sample output:**

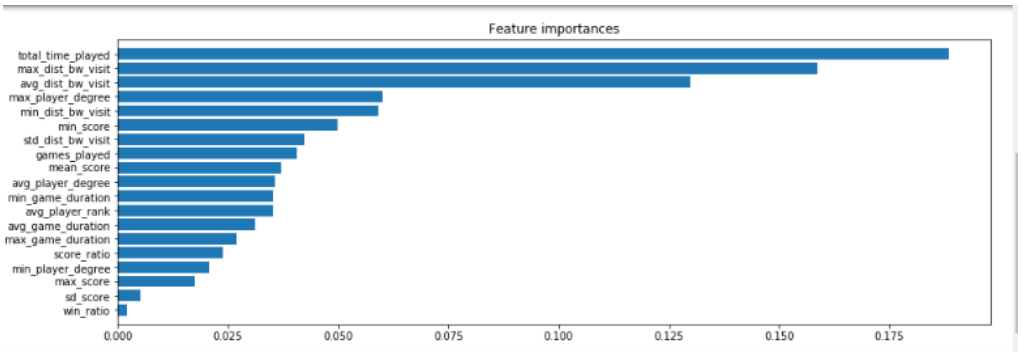
	0	1	2	3	4	5	6	7	8	9	...	231	232	233	234	235	236	237	238	239	churned
0	19	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	50	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
2	56	31	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

d. Sample Variable Importance Plots

1. Correlation Plot



2. Variable Importance Plot



V. Implementation

a. Parameter selection

1. **Deep learning models**
    1. **Models**
        a. Were selected based on the data, and optimized based on common best practices, experience, research and trial testing.
    2. **Observation period, churn period and Bin time**
        a. Determines how much data we use to train and how much will be used for determining the classification.
        b. selected based on the churn plot, training duration and end results of both supervised models and deep learning models.
        c. Shorter observation periods were tested to select the best observation period of our immediate churners and long observation periods for our gradual churners
    3. **Batchsize**
        a. Determines the size of input to be trained per epoch
        b. selected based on the GPU's capability and end results.
    4. **Kfold**
        a. Technique used to train and evaluate models by randomly partitioning original samples into k equal sized subsamples
        b. selected based on training duration.
    5. **Epochs**
        a. Technique used to determine the number of iterations to do on the model.
        b. Selected based on end results and the train vs. validation plot.
    6. **Scale data**
        a. Determines if we should scale our data inputs
        b. Selected based on model results rather than value distribution
2. **Supervised learning models**
    1. **Models**
        a. Selected based on the input data and optimized based on random grid search.
    2. **Observation period and churn period**
        a. Determines how much data we use to train and how much will be used for determining the classification.

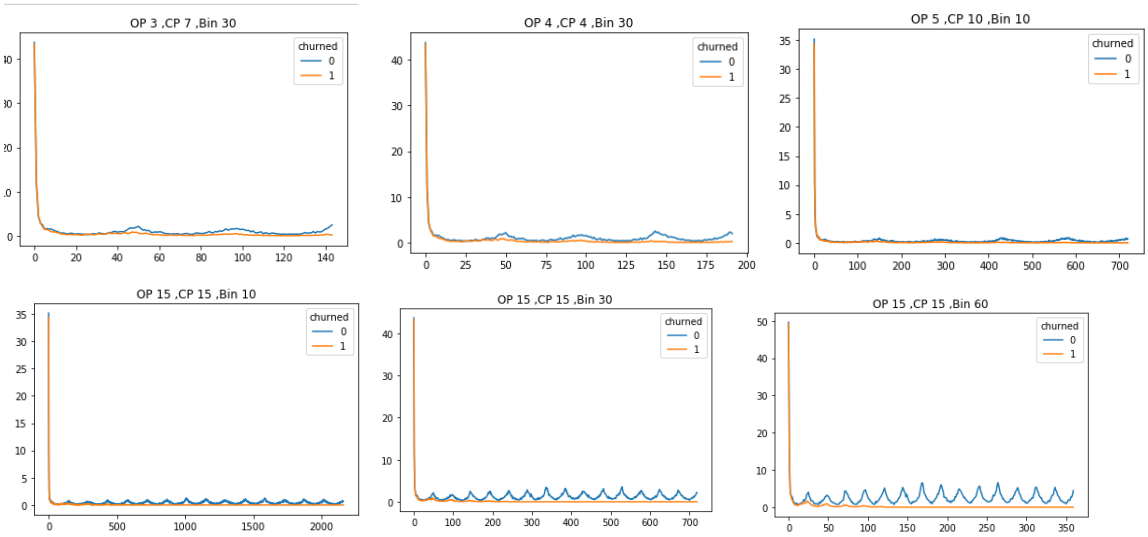
b. selected based on the churn plot, training duration and end results of both supervised models and deep learning models.
c. Shorter observation periods were tested to select the best observation period of our immediate churners and long observation periods for our gradual churners

3. **Include Team**
   a. Determines if we should include team data into our inputs.
   b. Selected based on feature selection
4. **Scale data**
   a. Determines if we should scale our data inputs
   b. Selected based on model results rather than value distribution

b. Parameter Selection Visualization

Churn plot

What was used to determine the observation period, churn period and bin period



VI. Models Results and Evaluation

a. Immediate churners

1. Top 3 Models

1. Logistic Regression

```
classifier:  LogisticRegression
accuracy score 0.7576896787423103
roc_auc_score 0.714667084563655
log loss score 8.36923700106705
zero_one_loss score 0.2423103212576897
confusion_matrix [[ 595  494]
 [ 215 1622]]
             precision    recall  f1-score   support

          0       0.73      0.55      0.63      1089
          1       0.77      0.88      0.82      1837

avg / total       0.75      0.76      0.75      2926
```

2. Gradient Boosting Classifier

```
classifier:  GradientBoostingClassifier
accuracy score 0.7665755297334245
roc_auc_score 0.7324002133474098
log loss score 8.062315004033149
zero_one_loss score 0.23342447026657553
confusion_matrix [[ 652  437]
 [ 246 1591]]
             precision    recall  f1-score   support

          0       0.73      0.60      0.66      1089
          1       0.78      0.87      0.82      1837

avg / total       0.76      0.77      0.76      2926
```

3. RNN Model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_7 (InputLayer)         (None, 192, 1)            0
_________________________________________________________________
cu_dnngru_14 (CuDNNGRU)      (None, 192, 256)          198912
_________________________________________________________________
dropout_14 (Dropout)         (None, 192, 256)          0
_________________________________________________________________
cu_dnngru_15 (CuDNNGRU)      (None, 192, 256)          394752
_________________________________________________________________
dropout_15 (Dropout)         (None, 192, 256)          0
_________________________________________________________________
cu_dnngru_16 (CuDNNGRU)      (None, 256)               394752
_________________________________________________________________
dropout_16 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_16 (Dense)             (None, 10)                2570
_________________________________________________________________
batch_normalization_7 (Batch (None, 10)                40
_________________________________________________________________
dense_17 (Dense)             (None, 1)                 11
_________________________________________________________________
activation_10 (Activation)   (None, 1)                 0
=================================================================
Total params: 991,037
Trainable params: 991,017
Non-trainable params: 20
_________________________________________________________________
accuracy score 0.7505126452494874
roc_auc_score 0.7242814646189715
log loss score 0.539649109494244
zero_one_loss score 0.24948735475051265
confusion_matrix [[ 677  412]
 [ 318 1519]]
             precision    recall  f1-score   support

          0       0.68      0.62      0.65      1089
          1       0.79      0.83      0.81      1837

avg / total       0.75      0.75      0.75      2926
```

4. Final model

I selected the best model from the previous section, gradient boosting classifier, then tuned it by doing a random grid search for selecting the hyperparameters. I also reduced the number of features based on the importance plot of the model.

```
Model with reduced feature space
classifier:  GradientBoostingClassifier
accuracy score 0.7655502392344498
roc_auc_score 0.7340140655328462
log loss score 8.097723730765253
zero_one_loss score 0.23444976076555024
confusion_matrix [[ 665  424]
 [ 262 1575]]
             precision    recall  f1-score   support

          0       0.72      0.61      0.66      1089
          1       0.79      0.86      0.82      1837

avg / total       0.76      0.77      0.76      2926
```

b. Gradual churners

1. Top 3 Models

1. RNN Model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
cu_dnnlstm_4 (CuDNNLSTM)     (None, 128)               67072
_________________________________________________________________
dense_18 (Dense)             (None, 1)                 129
_________________________________________________________________
activation_11 (Activation)   (None, 1)                 0
=================================================================
Total params: 67,201
Trainable params: 67,201
Non-trainable params: 0
_________________________________________________________________
accuracy score 0.9077238550922762
roc_auc_score 0.846892669316807
log loss score 0.23037450583250613
zero_one_loss score 0.09227614490772384
confusion_matrix [[ 306   95]
 [ 175 2350]]
             precision    recall  f1-score   support

          0       0.64      0.76      0.69       401
          1       0.96      0.93      0.95      2525

avg / total       0.92      0.91      0.91      2926
```

2. RNN Model 2

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
cu_dnnlstm_4 (CuDNNLSTM)     (None, 128)               67072
_________________________________________________________________
dense_18 (Dense)             (None, 1)                 129
_________________________________________________________________
activation_11 (Activation)   (None, 1)                 0
=================================================================
Total params: 67,201
Trainable params: 67,201
Non-trainable params: 0
_________________________________________________________________
accuracy score 0.9077238550922762
roc_auc_score 0.846892669316807
log loss score 0.23037450583250613
zero_one_loss score 0.09227614490772384
confusion_matrix [[ 306   95]
 [ 175 2350]]
             precision    recall  f1-score   support

          0       0.64      0.76      0.69       401
          1       0.96      0.93      0.95      2525

avg / total       0.92      0.91      0.91      2926
```

3. RNN Model 3

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_8 (InputLayer)         (None, 720, 1)            0
_________________________________________________________________
bidirectional_11 (Bidirectio (None, 720, 512)          397824
_________________________________________________________________
dropout_17 (Dropout)         (None, 720, 512)          0
_________________________________________________________________
bidirectional_12 (Bidirectio (None, 512)               1182720
_________________________________________________________________
dropout_18 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_19 (Dense)             (None, 10)                5130
_________________________________________________________________
batch_normalization_8 (Batch (None, 10)                40
_________________________________________________________________
dense_20 (Dense)             (None, 1)                 11
_________________________________________________________________
activation_12 (Activation)   (None, 1)                 0
=================================================================
Total params: 1,585,725
Trainable params: 1,585,705
Non-trainable params: 20
_________________________________________________________________
accuracy score 0.8892686261107313
roc_auc_score 0.8666166267499568
log loss score 0.2845036996418417
zero_one_loss score 0.11073137388926868
confusion_matrix [[ 335   66]
 [ 258 2267]]
             precision    recall  f1-score   support

          0       0.56      0.84      0.67       401
          1       0.97      0.90      0.93      2525

avg / total       0.92      0.89      0.90      2926
```

4. Final Model

I selected the best model from the previous section, RNN Model 3. I tried tuning it by doing a K-fold cross validation, but the model did not improve, most notably because deep learning models are not able to tune directly with ROC AUC score. So the final model is better of using RNN Model 3 as it is.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_8 (InputLayer)         (None, 720, 1)            0
_________________________________________________________________
bidirectional_11 (Bidirectio (None, 720, 512)          397824
_________________________________________________________________
dropout_17 (Dropout)         (None, 720, 512)          0
_________________________________________________________________
bidirectional_12 (Bidirectio (None, 512)               1182720
_________________________________________________________________
dropout_18 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_19 (Dense)             (None, 10)                5130
_________________________________________________________________
batch_normalization_8 (Batch (None, 10)                40
_________________________________________________________________
dense_20 (Dense)             (None, 1)                 11
_________________________________________________________________
activation_12 (Activation)   (None, 1)                 0
=================================================================
Total params: 1,585,725
Trainable params: 1,585,705
Non-trainable params: 20
_________________________________________________________________
accuracy score 0.8892686261107313
roc_auc_score 0.8666166267499568
log loss score 0.2845036996418417
zero_one_loss score 0.11073137388926868
confusion_matrix [[ 335   66]
 [ 258 2267]]
             precision    recall  f1-score   support

          0       0.56      0.84      0.67       401
          1       0.97      0.90      0.93      2525

avg / total       0.92      0.89      0.90      2926
```

VII. Results summary:

As we can see from our results, there is a huge difference between determining immediate churners and gradual churners. The task of identifying immediate churners is not easy, as we can see from our best model which attained a score of 0.73 UAC ROC. The score is not so bad considering the amount of data we have and the short observation period. On the other hand, determining gradual churners was an easier task as it considers a lot more data. Our results are also much better, with a UAC score of 0.87 from a deep learning model. One interesting aspect we can see from this is that supervised learning classifiers do better when we use little data but are no where near to deep learning classifiers when we have more data. All in all, I believe that we achieved great results for our problem. This solution may not be the best solution out there, but it is able to determine players that are either likely to quit immediately and gradually.

VIII. Conclusion

a. Approach

This may serve as a summary of the entire process I used for this project

1. Find the definition of churn and research on the topic
2. Identify the objective of the project
3. Understand the data input
4. Analyze the data obtained
5. Research similar problems and their solutions
6. List down and analyze potential solutions (preprocessing and model) that can work for the churning problem based on our data.

7. Select the approach and solutions based on the project objective, data, case scenario and deadline
8. Start working on this paper
9. List tasks and set deadline for each task
10. Preprocess data
11. Implement a simple solution
12. Adjust feature selection
13. Test new solution
14. Identify problem areas that stop the implementation from reaching desired results.
15. Identify opportunities/solutions to the problems listed
16. Apply most effective solution
17. Simplify code
18. Iterate steps 11 to 17 until satisfied with results
19. Update paper
20. Try working on a new solution, approach or model.

b. Improvement

There are some improvements or solutions that I would have loved to try and work on if I had the time. The following are improvements and solutions that would have worked:

1. **Improvements:**
    1. PCA to combine features that correlate with each other and reduce computation cost.
    2. Using distributed computing tools like spark to build a preprocessing pipeline
    3. Model ensembles to attain better results by combining models together.
    4. Adding more visualizations for our data, process and results. This is to make it easier to understand the reasoning behind parameter selection or tuning.
    5. Use a more valid way of selecting parameters, like using sklearn 's RFECV(Recursive feature elimination with cross-validation)
    6. Obtaining more data to have our models produce better results.
2. **Solutions:**
    1. Using of clustering algorithms to identify churned users.
    2. Clustering users together by their activity patterns to make it possible to work on multivariate time series prediction.
    3. Work on time series predictions.
    4. Try existing solutions such as WTTE-RNN to identify churners.