MusicBox App Churn Analysis & Prediction

Tianyi Fang
03/18/2018

BITTIGER

The Lifelong Learning Platform of Silicon Valley

Outline

BITTIGER

- Pre-process Data Preparation
- Exploratory Data Analysis
- Feature Engineering
- Prediction Model Evaluations
- Improvement & Recall

Pre-process Data Preparation

BITTIGER

- Loading --Spark
- Cleaning
  - play_time:
  - song_length
  - robot user(outliers:>5400(99%)
  - song_id
- Extracting
  - Drop song_name/singer..etc
- Churn Label Definition
  - Based on play.log
    (Churn if no play after 4/29)
- Down-sampling
  - Balance Churn/Active

```
uid_label.groupBy('Churn').count().show()
```

```
+-----+-----+
|Churn|count|
+-----+-----+
|    1|51419|
|    0|89216|
+-----+-----+
```

```
sampled_uid = uid_label.sampleBy('Churn', fractions = {1:0.086, 0:0.05})
sampled_uid.groupBy('Churn').count().show()
```

```
+-----+-----+
|Churn|count|
+-----+-----+
|    1| 4358|
|    0| 4504|
+-----+-----+
```

```
def parseLine(line):
    fields = line.split("\t")
    if len(fields) == 10:
        try:
            uid = float(fields[0])
            device = str(fields[1])
            song_id = float(fields[2])
            song_type = float(fields[3])
            song_name = str(fields[4])
            singer = str(fields[5])
            play_time = float(fields[6])
            song_length = float(fields[7])
            paid_flag = float(fields[8])
            file_name = str(fields[9])
            return Row(uid, device, song_id, song_type, song_name, singer,
                       play_time, song_length, paid_flag, file_name)
        except:
            return Row(None)
    else:
        return Row(None)
```

```
schema = StructType([StructField('uid', FloatType(), False),
                     StructField('device', StringType(), True),
                     StructField('song_id', FloatType(), False),
                     StructField('song_type', FloatType(), True),
                     StructField('song_name', StringType(), True),
                     StructField('singer', StringType(), True),
                     StructField('play_time', FloatType(), False),
                     StructField('song_length', FloatType(), True),
                     StructField('paid_flag', FloatType(), True),
                     StructField('file_name', StringType(), True),])
```

Pre-process Data Preparation

BITTIGER

My cleaning principle is:

Assumption: each record only record one play time(not cumulative), if user play one song iteratively, the records will show multiple records, thus play_time should be ≤ song_length

play_time:

- if >10E5(only (20/4235452) no sense)(user = 17) or play_time == 0(user skipped play or clicked by error), drop them(user =
- if >10E4, devided by 1000. Since it's possible play_time was recorded as miliseconds
- if > song_length, check its correctness, impute by the above way or song_length-1

song_length:

- if >10E6(no sense) or = 0, impute with play_time this step is done in Spark; by later check, find there is no song_length >10E4, which means existed song_length are reasonable, and imputed song_length has reasonable play_time. Play_time can be derived by song_length

[Notice]song_id:

- song_id ==0, with multiple song_length and play_time by different users, which means it's not a unique song, it's happened possibly due to mislocation of song_type(with value=0,1,2). There are nearly 10%(346692/4235452) of song_id =0. It's OK to just leave them as original, since we are now doing Churn prediction. However, if we want to do song_recommendation in the future, we should find better way to impute them.
- song_id = -1(177/4235452), with 9 users. Also keep.

Summary of the amount change of user

original user is 141644

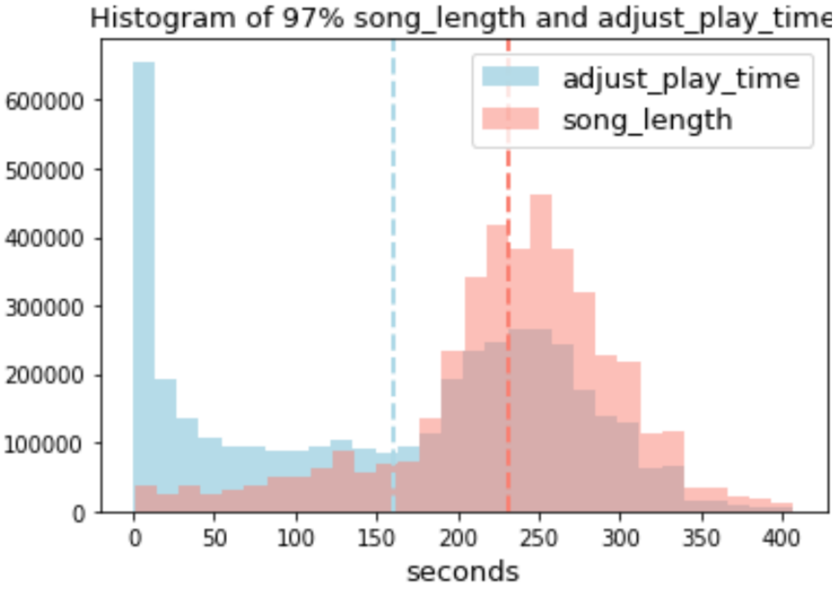
valid user is 140635 with **active user** 89216, **churn user** 51419

downsampled user is 8862, with **active** 4504, **churn** 4358

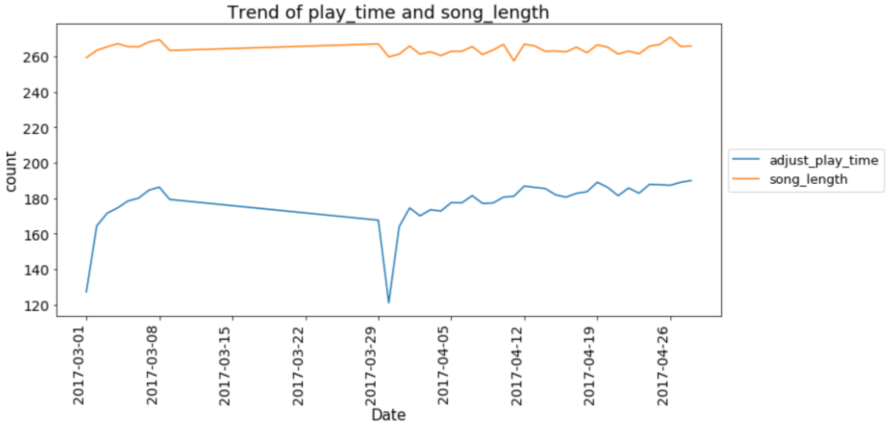
	uid	song_id	date
24	49423096	4348548.0	2017-04-05
25	49423096	4348548.0	2017-04-05
77	49423096	4348548.0	2017-04-06
78	49423096	4348548.0	2017-04-06
151	49423096	4348548.0	2017-04-07
4955453	168915904	6611804.0	2017-04-27
4955454	168915904	6611804.0	2017-04-27

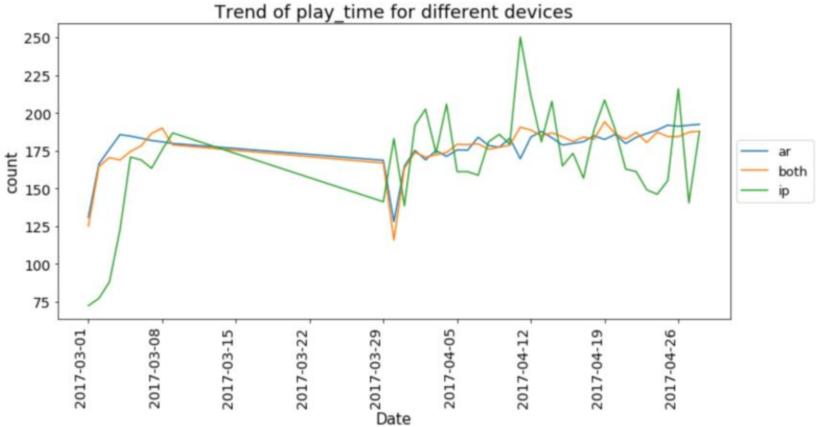
Exploratory Data Analysis

• play_time/song_length

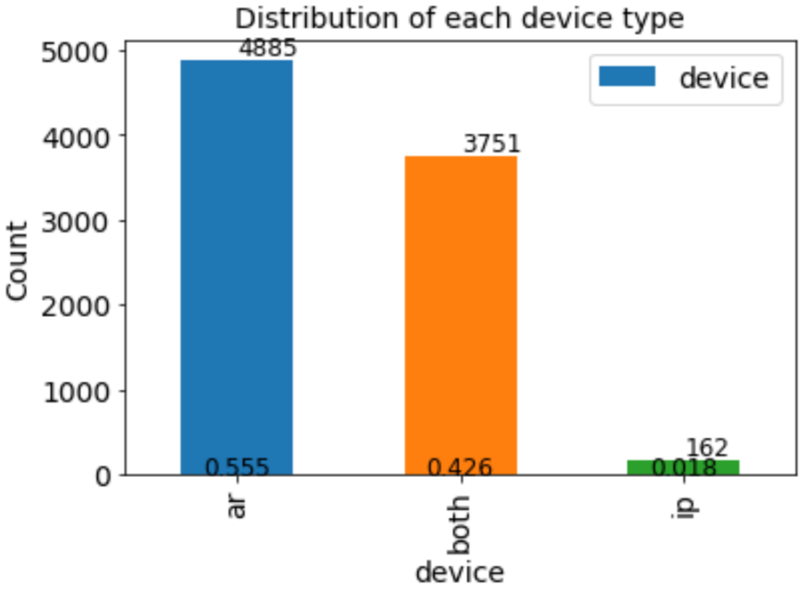


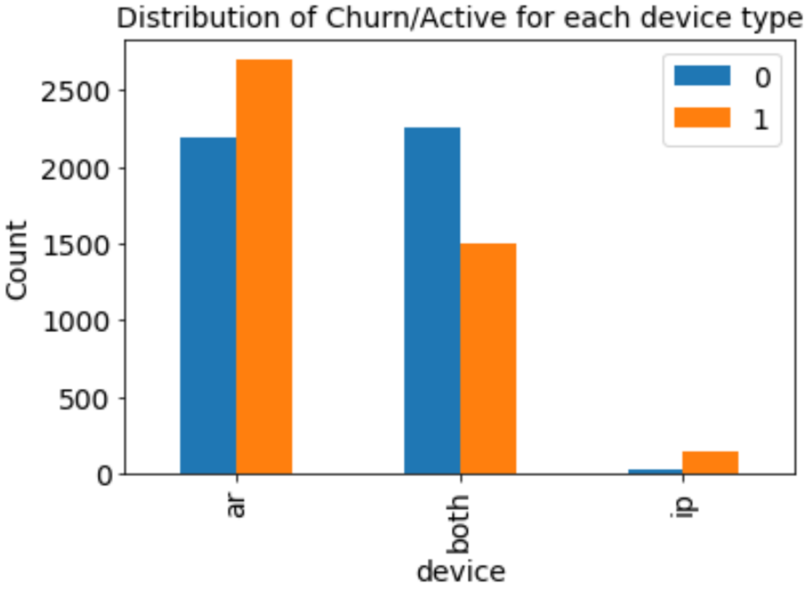
• trend





Exploratory Data Analysis

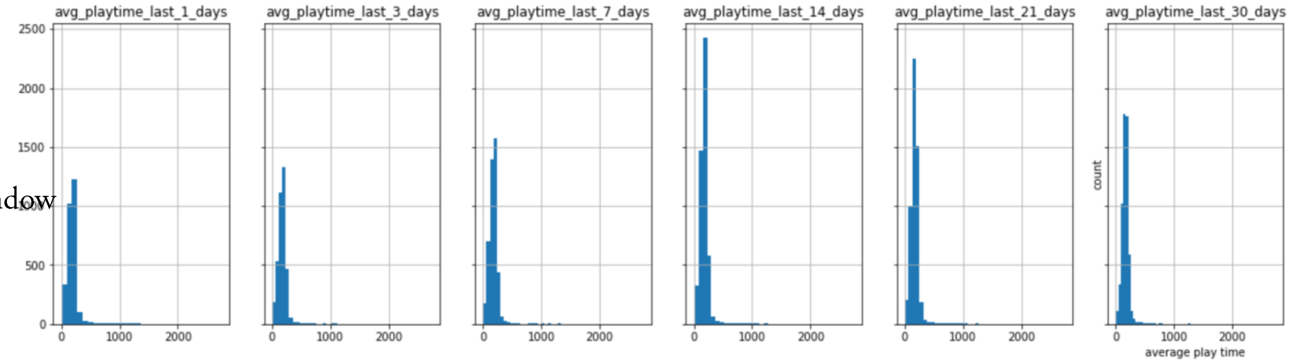


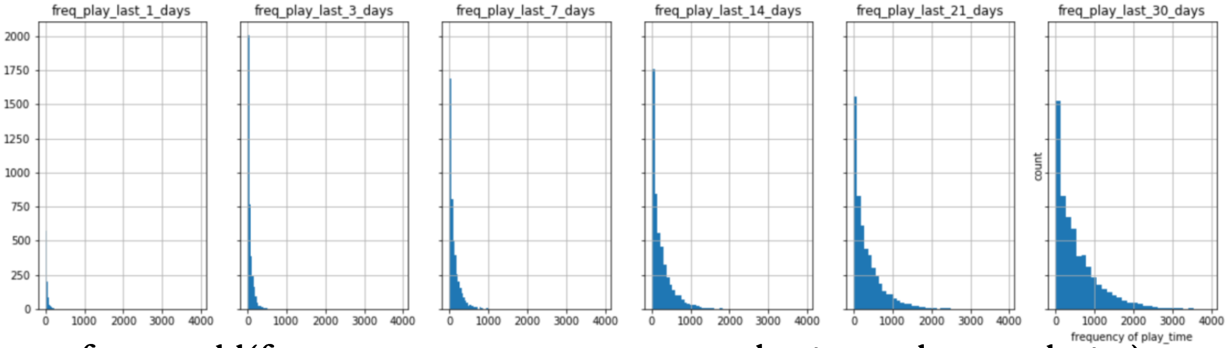


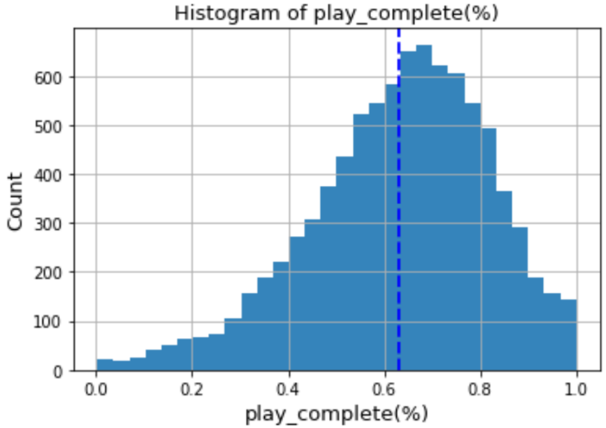
Feature Engineering

BITTIGER

- Frequency
  - Time window =[1,3,7,14,21,30]
  - Events = [‘play’, ‘search’, ‘download’]
- Average playtime within time window
  - Sum(play_time in time_window)/time_window
- Recency
- Play complete degree
  - avg(play_time/song_length) for each user







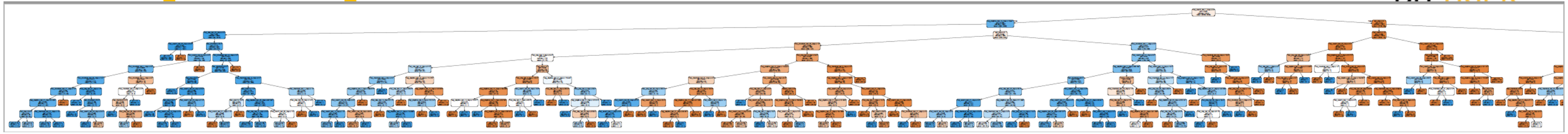
- feature_dd(frequency + recency + average_playtime + dummy device)
- feature_total(feature_dd + complete precent)

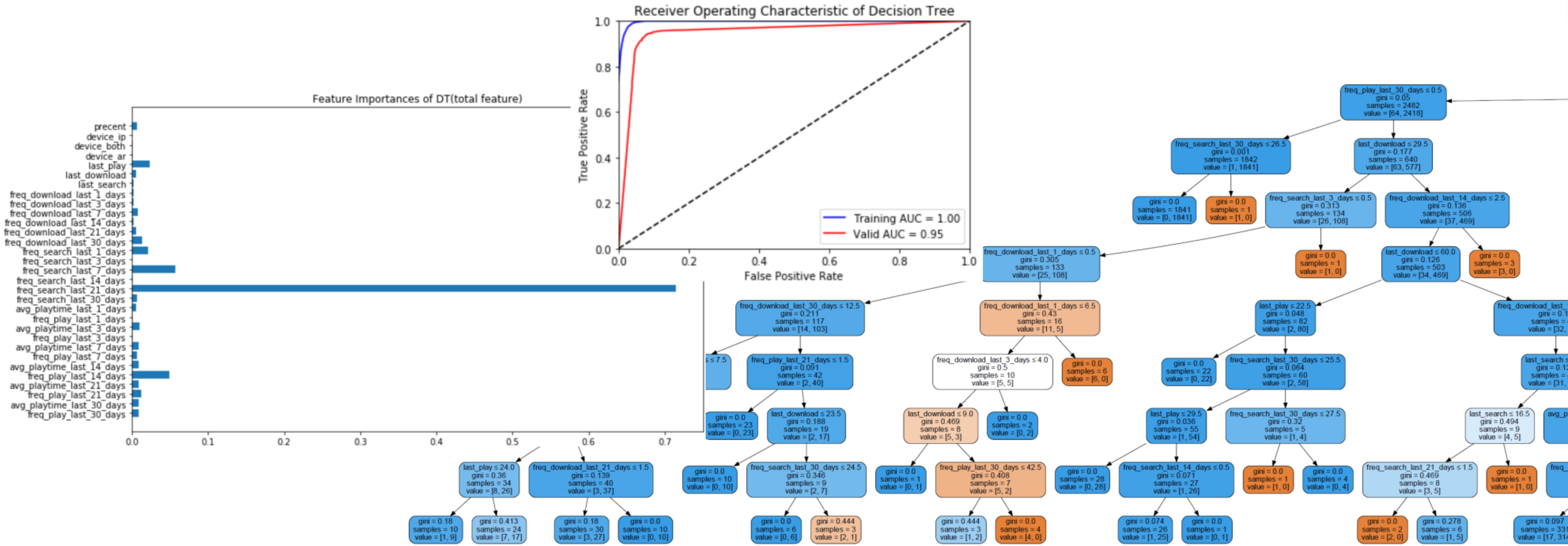
Prediction Model

		parameters	Best_param_evaluate
Decision Tree	DecisionTreeClassifier(criterion='gini',max_depth =None, max_features =None,max_leaf_nodes=None,min_impurity_decrease=0.0,min_impurity_split=None,min_samples_leaf=1,min_samples_split=2, splitter='best')	criterion='gini' max_depth =10 max_features =5 min_samples_leaf=1 min_samples_split=2	0.944373605355435 1 {'max_depth': 6, 'max_features': 5}
Random Forest	RandomForestClassifier(bootstrap=True, criterion='gini', max_depth=None, max_features='auto', max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=2, n_estimators=10, n_jobs=1, oob_score=False, random_state=None, verbose=0, warm_start=False)	bootstrap=True criterion='gini' **max_depth=15** max_features='sqrt' **min_samples_leaf=7** **min_samples_split=3** **n_estimators=50** oob_score=True	0.907363136504900 7 {'max_depth': 15, 'min_samples_leaf': 7, 'min_samples_split': 3, 'n_estimators': 50}
AdaBoost	AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0, n_estimators=50, random_state=None)	algorithm='SAMME.R' base_estimator=adaboost **learning_rate=0.2** **n_estimators=50**	0.989126554709455 3 {'learning_rate': 0.2, 'n_estimators': 50}

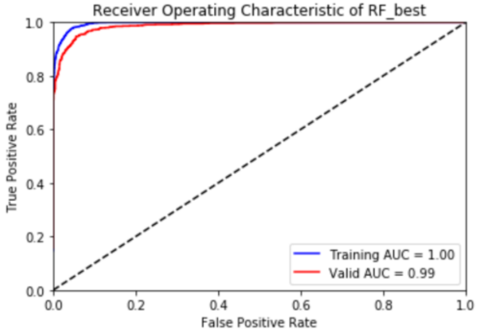
Default Model → RandomizedSearchCV → GridSearchCV → Best Model

Prediction Model Tree





Prediction Model Confusion Matrix



feature_dd

Decision Tree

	Predicted Active	Predicted Churn
True Active	1021	77
True Churn	72	1030

Adaboost

	Predicted Active	Predicted Churn
True Active	1035	63
True Churn	64	1038

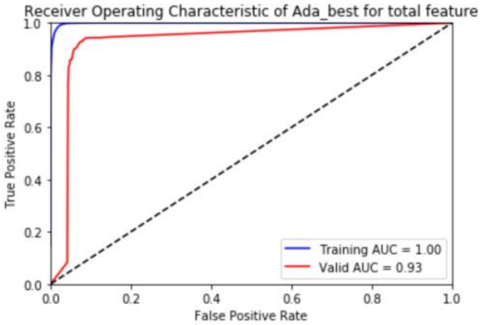
feature_total

Decision Tree

	Predicted Active	Predicted Churn
True Active	1038	83
True Churn	91	988

Adaboost

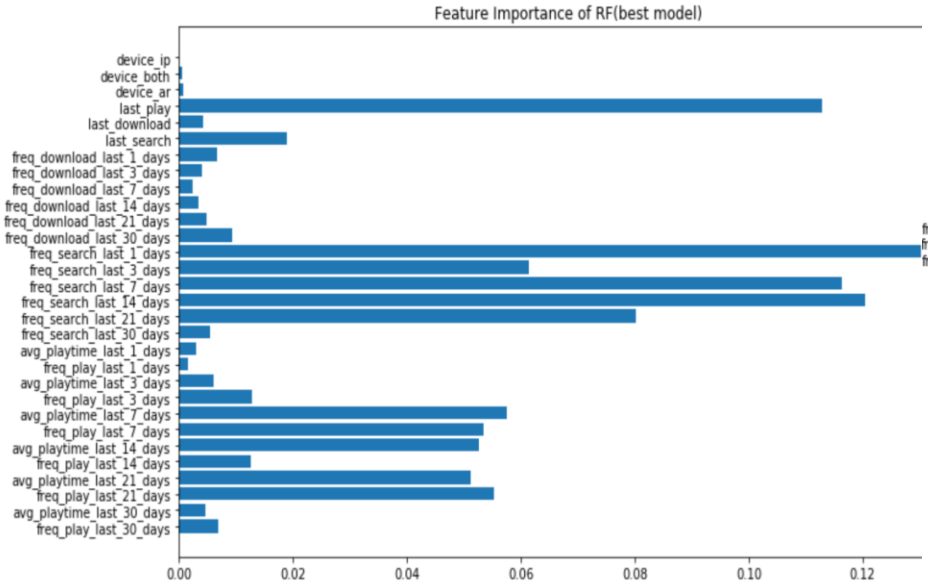
	Predicted Active	Predicted Churn
True Active	1060	61
True Churn	73	1006

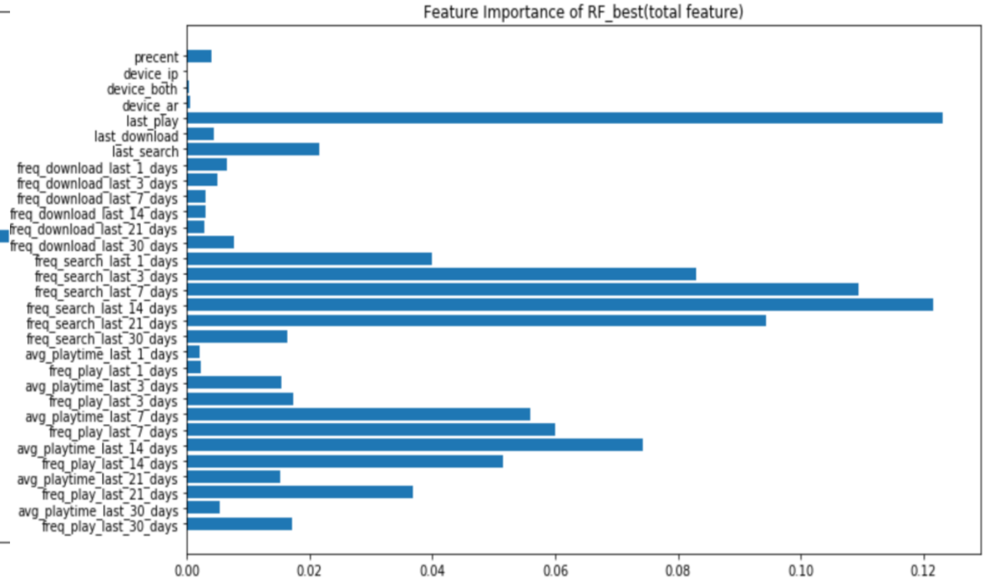


	train_dt	valid_dt	train_ada	valid_ada	train_ada_best	valid_ada_best	train_dt_total	valid_dt_total	train_ada_total	valid_ada_total	train_ada_total_best	valid_ada_total_best
Accuracy	0.974498	0.930688	0.948996	0.933556	0.946605	0.936902	0.983264	0.926386	0.949793	0.940727	0.943736	0.940727
Precision	0.958923	0.916746	0.940288	0.931170	0.941502	0.935039	0.975635	0.919355	0.939507	0.935354	0.936033	0.938900
Recall	0.986833	0.942913	0.949411	0.932087	0.942481	0.935039	0.988683	0.924949	0.953361	0.939148	0.943416	0.935091
f1_score	0.972678	0.929646	0.944828	0.931628	0.941991	0.935039	0.982115	0.922144	0.946383	0.937247	0.939710	0.936992
AUC	0.975412	0.931029	0.949027	0.933515	0.946300	0.936851	0.983621	0.926308	0.950028	0.940641	0.943715	0.940421
Matthews_	0.949121	0.861704	0.897444	0.867007	0.892531	0.873701	0.966466	0.852349	0.899268	0.881094	0.886994	0.881042

- feature_dd(frequency + recency + average_playtime + dummy device)
- feature_total(feature_dd + complete precent)

Prediction Model Feature Importance

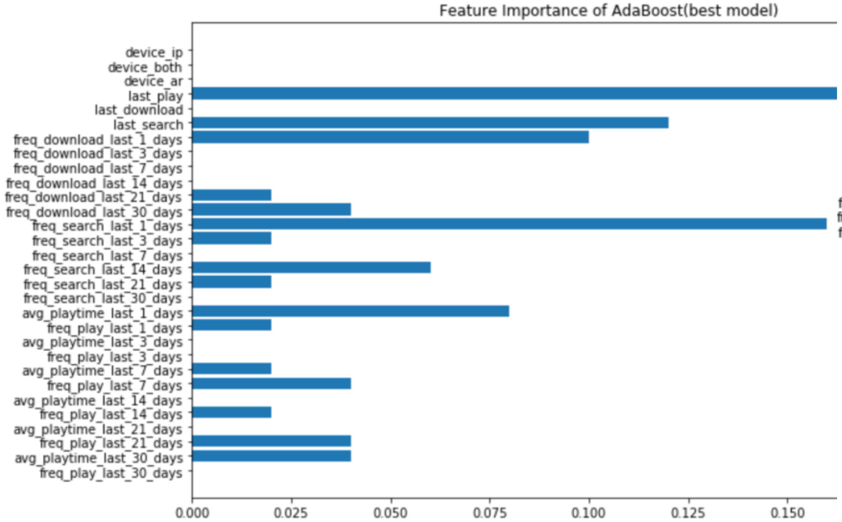


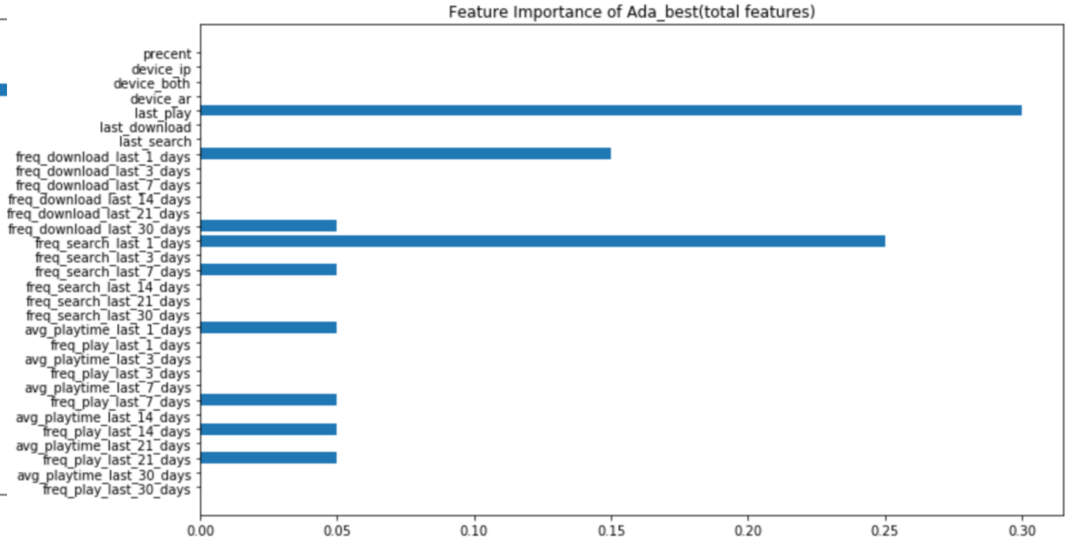


'freq_search_last_1_days' **'freq_search_last_14_days'**
'freq_search_last_7_days' **'last_play'**
'freq_search_last_21_days' 'freq_search_last_3_days'
'avg_playtime_last_7_days' 'freq_play_last_21_days'
'freq_play_last_7_days'

'last_play' **'freq_search_last_14_days'**
'freq_search_last_7_days' 'freq_search_last_21_days'
'freq_search_last_3_days' 'avg_playtime_last_14_days'
'freq_play_last_7_days' **'avg_playtime_last_7_days'**
'freq_play_last_14_days'

Prediction Model Feature Importance





'last_play' 'freq_search_last_1_days'
'last_search' **'freq_download_last_1_days'**
'avg_playtime_last_1_days'
'freq_search_last_14_days'
'freq_play_last_21_days'
'freq_download_last_30_days'
'freq_play_last_7_days'

'last_play' 'freq_search_last_1_days'
'freq_download_last_1_days' 'freq_play_last_7_days'
'avg_playtime_last_1_days'
'freq_download_last_30_days'
'freq_search_last_7_days' 'freq_play_last_14_days'
'freq_play_last_21_days'

Prediction Model Feature Importance

BITTIGER

```
'last_play’
'freq_search_last_1_days’
'freq_play_last_21_days’
'avg_playtime_last_1_days’
'avg_playtime_last_7_days'
```

Improvement & Recall of Churn User

BITTIGER

- Specify Main Business Goal for Churn(**prevent churn/recall churn**)

  Refined Definition of Churn: **churn_type** based on key action/ time_window..
- **User portfolio**(region, age, gender, membership, survey, email, …)
- Better **technical support**(accurate data provides more useful information)

  platform/account/API/version/latency
- Churn Reason Analysis(AARRR& survey)

  Product? Design? Function? Operation? Competitor? Availability?
- Improvement
- **user segmentation(model/ business insights)**>> model: impute data/ more features

  potential churn user/churned user: promotion push, coupon, ads

Thank You for Joining Us!

BITTIGER

The Lifelong Learning Platform of Silicon Valley