Relax Product Offering

Relax offers an online product service. To use the service a customer creates an account and then can start to use the service. The company would like to determine which factors predict future user adoption. To do this, I have been supplied with two files. The layouts are as follows:

1] A user table (*"takehome_users"*) with data on 12,000 users who signed up for the product in the last two years.

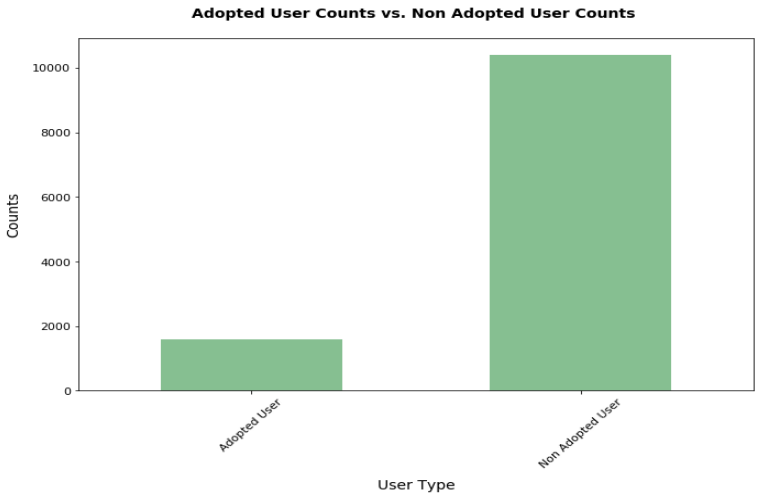
- **name:** the user's name
- **object_id:** the user's id
- **email:** email address
- **creation_source:** how their account was created. This takes on one of 5 values:
    - **PERSONAL_PROJECTS:** invited to join another user's personal workspace
    - **GUEST_INVITE:** invited to an organization as a guest (limited permissions)
    - **ORG_INVITE:** invited to an organization (as a full member)
    - **SIGNUP:** signed up via the website
    - **SIGNUP_GOOGLE_AUTH:** signed up using Google Authentication (using a Google email account for their login id)
- **creation_time:** when they created their account
- **last_session_creation_time:** UNIX timestamp of last login
- **opted_in_to_mailing_list:** whether they have opted into receiving marketing emails
- **enabled_for_marketing_drip:** whether they are on the regular marketing email drip
- **org_id:** the organization (group of users) they belong to
- **invited_by_user_id:** which user invited them to join (if applicable).

2] A usage summary table (*"takehome_user_engagement"*) that has a row for each day that a user logged into the product.

Adoption is defined as a user who has logged into the product on three separate days in at least one seven-day period.

There are two columns that have missing values, last_session_creation_time and invited_by_user_id. The fields were filled with 0 values.

The following is a plot of the classes (adopted user and non-adopted user)



Note the class imbalance between the adopted user and non-adopted user. Discussed will be techniques to deal with this situation

Relax Product Offering

Data was prepped for machine learning. Labels and features were created, and train/test split was performed at 70% training, 30% testing. I had to laugh on my first run, I got an accuracy of .87 but there were 0 true positives. It was because of class imbalance. I was missing a feature last_session_creation_time which I normalized and added to the model. I then got more true positives, but recall suffered due to imbalance.

For the model, I used the random forests and extreme gradient boosting. I also used the class_weight parameter for random forests and scale_pos_weight for extreme gradient boosting.

Results without weighted classes are as follows:

Random Forests

```
Accuracy on training set is : 0.9255952380952381
Accuracy on test set is : 0.9236111111111112
              precision    recall  f1-score   support

           0       0.73      0.67      0.70       476
           1       0.95      0.96      0.96      3124

   micro avg       0.92      0.92      0.92      3600
   macro avg       0.84      0.81      0.83      3600
weighted avg       0.92      0.92      0.92      3600

[[ 317  159]
 [ 116 3008]]
```

Feature Importances:

	name	coef
2	nlast_session_creation_time	0.979626
5	cs_PERSONAL_PROJECTS	0.009175
3	cs_GUEST_INVITE	0.002823
7	cs_SIGNUP_GOOGLE_AUTH	0.002481
1	enabled_for_marketing_drip	0.002093
0	opted_in_to_mailing_list	0.001969
4	cs_ORG_INVITE	0.001029
6	cs_SIGNUP	0.000805

XGB Classifier

```
Accuracy on training set is : 0.9352380952380952
Accuracy on test set is : 0.9222222222222223
              precision    recall  f1-score   support

           0       0.74      0.63      0.68       476
           1       0.94      0.97      0.96      3124

   micro avg       0.92      0.92      0.92      3600
   macro avg       0.84      0.80      0.82      3600
weighted avg       0.92      0.92      0.92      3600

[[ 300  176]
 [ 104 3020]]
```

Feature Importances:

	name	coef
2	nlast_session_creation_time	0.731690
5	cs_PERSONAL_PROJECTS	0.072060
3	cs_GUEST_INVITE	0.038984
7	cs_SIGNUP_GOOGLE_AUTH	0.038593
1	enabled_for_marketing_drip	0.034882
4	cs_ORG_INVITE	0.032071
6	cs_SIGNUP	0.027626
0	opted_in_to_mailing_list	0.024093

Using class_weight for random forests and scale_pos_weight for extreme gradient boosting, I get

Random Forests

```
Accuracy on training set is : 0.9234523809523809
Accuracy on test set is : 0.9205555555555556
              precision    recall  f1-score   support

           0       0.70      0.70      0.70       476
           1       0.95      0.95      0.95      3124

   micro avg       0.92      0.92      0.92      3600
   macro avg       0.83      0.83      0.83      3600
weighted avg       0.92      0.92      0.92      3600

[[ 335  141]
 [ 145 2979]]
```

Feature Importances:

	name	coef
2	nlast_session_creation_time	0.976147
5	cs_PERSONAL_PROJECTS	0.011391
3	cs_GUEST_INVITE	0.003382
7	cs_SIGNUP_GOOGLE_AUTH	0.002908
1	enabled_for_marketing_drip	0.002017
0	opted_in_to_mailing_list	0.002012
4	cs_ORG_INVITE	0.001344
6	cs_SIGNUP	0.000799

XGB Classifier

```
Accuracy on training set is : 0.9265476190476191
Accuracy on test set is : 0.92
              precision    recall  f1-score   support

           0       0.70      0.70      0.70       476
           1       0.95      0.95      0.95      3124

   micro avg       0.92      0.92      0.92      3600
   macro avg       0.83      0.83      0.83      3600
weighted avg       0.92      0.92      0.92      3600

[[ 332  144]
 [ 144 2980]]
```

Feature Importances:

	name	coef
2	nlast_session_creation_time	0.776342
5	cs_PERSONAL_PROJECTS	0.067456
3	cs_GUEST_INVITE	0.039799
7	cs_SIGNUP_GOOGLE_AUTH	0.034451
6	cs_SIGNUP	0.024329
1	enabled_for_marketing_drip	0.021633
0	opted_in_to_mailing_list	0.020548
4	cs_ORG_INVITE	0.015442

As you can see, the weights balance the classes out somewhat. As alternatives, you can under sample the true negatives (large class) or over sample true positive (small class) using repetition, bootstrapping or SMOTE (Synthetic Minority Over-Sampling Technique) techniques.

Regardless, of class imbalance, the normalized “last_session_creation_time” was by far the biggest factor in predicting adoption. The feature importances are listed above. Using just the normalized “last_session_creation_time” resulted in an accuracy of .92 for both algorithms.