Relax Challenge

Data Wrangling

In this section I did some exploration of the features available. We have 12000 different users, but in the **user_engagement** data there are only 8823 unique users, therefore more than 3000 of registeres users never visited the website.

```
adopted_users = []
for user in user_ids:
    user_df = df[df['user_id'] == user]
    if user_df.resample('W').count().visited.max() >= 3:
        adopted_users.append(user)
```

```
len(adopted_users)
```

```
1445
```

The next step was to create the adopted_users feature using the users' engagement.
We have 1445 users that had 3 ore more logins within a 7-day time span.
We used this to create a Target Feature, **adopted,** in our **users'** database.

Feature Engineering

1. I used **last_session_creation_time** to create **time_since_last_sesion** and **at_least_1_visit** features:
Time since last visit is simply the difference between the amx timestamp and the date of last sesion
At leat 1 visit is a binomial feature, 0 for those who had no visit records, and 1 for the rest.
2. Using **creation_time** to create **creation_year, creation_month** and **creation_dayofweek**
3. Lastly, I used **invited_by_user_id** to create **number_of_members_invited** feature with a simple groupby and left merge on users DataFrame.
After the removal of unnecesary feature, this is the data before modeling.

```
users.head().T
```

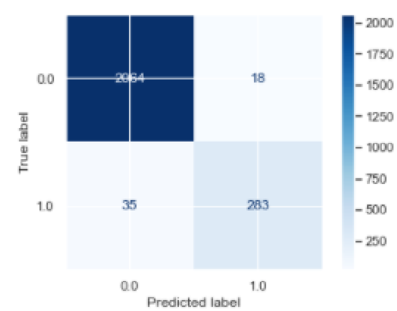
	0	1	2	3	4
creation_source	GUEST_INVITE	ORG_INVITE	ORG_INVITE	GUEST_INVITE	GUEST_INVITE
opted_in_to_mailing_list	1	0	0	0	0
enabled_for_marketing_drip	0	0	0	0	0
org_id	11	1	94	1	193
adopted	0	1	0	0	0
time_since_last_session	3.92792e+06	5.82923e+06	3.83318e+07	3.28566e+07	4.32171e+07
at_least_1_visit	1	1	1	1	1
creation_year	2014	2013	2013	2013	2013
creation_month	4	11	3	5	1
creation_dayofweek	1	4	1	1	3
number_of_members_invited	0	0	1	0	0

Modeling

I used 3 classifiers: XGBoost, RFC and KNN. XGBoost had the highest score with 0.97 accuracy and 0.99 ROC, folowed closely by RFC.

```
results.sort_values('ROC_auc', ascending = False)
```

	model	Accuracy	ROC_auc
1	XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,\n colsample_by...	0.9779	0.9929
0	(DecisionTreeClassifier(max_features='auto', random_state=1262095510), DecisionTreeClassifier(ma...	0.9750	0.9895
6	KNeighborsClassifier(n_neighbors=22)	0.9058	0.9365
7	KNeighborsClassifier(n_neighbors=30)	0.8988	0.9340
5	KNeighborsClassifier(n_neighbors=16)	0.9129	0.9315



The very high scores are ussualy explained by the fact that there might be some data leackeage in our features.

Using Shap library we can identify the most important features for our XGBoost model. It seems that time_since_last_sesion, creatin_year and creation_month are the main features in our prediction. So all we need to make a somewhat accurate predcition are the creation date and sime since last session to predict **adoption.**