The problem is to define an "adopted user", a user who has logged into the product on three separate days in at least one sevenday period and identify the factors that predict user adoption.We are given two data files; ‘takehome_user_engagement. csv’ and ‘takehome_users.csv’.

Dataframe ‘takehome_users.csv’ contain data on 12000 users who signed up for the product. It contains multiple column features. Dataframe corresponding to data ‘takehome_user_engagement.csv’ has a row for each day that a user logged into the product. Before beginning the modeling, the major task to perform is data wrangling and EDA.

If I denote data ‘takehome_users.csv’ by dataframe called **summ**, I did following to create the target features.

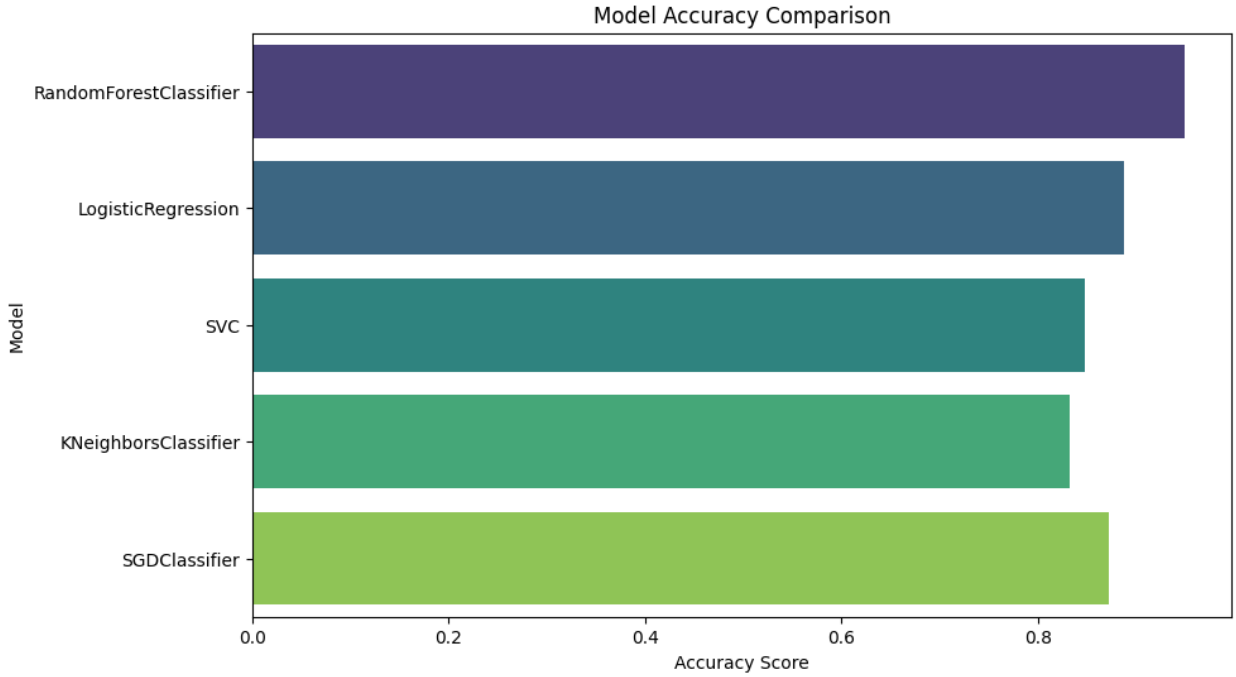
```
from datetime import timedelta
import pandas as pd
summ['time_stamp'] = pd.to_datetime(summ['time_stamp'])
summ_sorted = summ.sort_values(['user_id', 'time_stamp'], ignore_index=True)
summ_sorted['time_diff'] = summ_sorted.groupby('user_id')['time_stamp'].diff()
summ_sorted['time_diff'] = summ_sorted['time_diff'].dt.total_seconds()
summ_sorted['adopted_user'] =
summ_sorted.groupby('user_id')['time_diff'].apply(
    lambda group: any(group.rolling(window=3, min_periods=1).sum() <=
7*24*3600)  # 7 days in seconds
).reset_index(level=0, drop=True)
```

After doing this, we should clean the dataframe, deal with missing values, convert categorical columns into numeric, do some feature engineering, and group dataframe inorder to have 12000 rows for each user_id and corresponding 12000 entries for the target column ‘adopted_user’.

Next, we merge this dataframe (summ) with dataframe obtained from data ‘takehome_user_engagement.csv’. Again, we need to do data wrangling and EDA steps for theis merged dataframe which we say **df**.

Next, I started modeling. I splitting the data into train and test sets in 8:2 ratio. I tried RandomForestClassifier, LogisticRegression, support vector machine classifier, KNeighborsClassifier, and SGDClassifier. For each models I evaluated the model accuracy suing accuracy_score metric. It turns out that Random Forest model stands

out with accuracy score of 0.9492. The performance of different models can be visualized as follow;



Next, to further tune the random forest model, I proceeded with the hyperparameter tuning. I used the gridsearch method with cross validation as shown by line below; grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid, scoring='accuracy', cv=5). The performance of this model is also understood by following classification report;

	precision	recall	f1-score	support
0.0	0.96	0.99	0.97	2035
1.0	0.90	0.75	0.82	365
accuracy			0.95	2400
macro avg	0.93	0.87	0.89	2400
weighted avg	0.95	0.95	0.95	2400

Tuning the model helped to increase the accuracy of model from 0.9492 to 0.9496 by 0.04 % only.

At last, I did the feature importance test, and found that column ‘account_age_days’ is most importance with 36.6 % importance. ‘Account_age_days’ is actually the account hold duration by users (difference between last_session_creation_time and account creation time).