# User Adoption Prediction: Data Analysis and Insights

I performed data cleaning and transformation, including handling missing values and converting relevant columns to datetime objects for further analysis. Standard data preprocessing procedures were followed.

We defined "adopted users" based on the criteria mentioned in the task. A user is adopted if they have logged in on three separate days within a seven-day period. Was created function to define them.

```
# Define a function to check if a user is adopted
def is_adopted(user_data):

    #Create DataFrame for marking users adopted (1) or not (0)
    columns = ['user_id', 'is_adopted']
    df = pd.DataFrame(columns=columns)

    #Loop through unique user_id
    for j in user_data['user_id'].unique():

        #Find login timestamp for each user
        user_time = user_data[user_data['user_id'] == j]

        #Find difference between last and first login for the user
        last_first_diff = user_time.iloc[-1]['time_stamp'] - user_time.iloc[0]['time_stamp']

        #Check if difference between last and first login for the user is less or equal 7 days
        if last_first_diff <= pd.Timedelta(days=7):

            #Find sum of times user logged in during this period
            s = sum(user_time['visited'])

            #If user logged in 3 or more times then mark him as adopted
            if s>=3:
                adopted = 1
            else:
                adopted = 0
```

```
        else: #If difference between last and first login for the user is more then 7 days

            s=0

            for i in range(len(user_time['time_stamp'])-2):

                #Find 7 days window starts with each login except last 2
                window = user_time.iloc[i]['time_stamp'] + pd.Timedelta(days=7)

                #Find the third login for each login
                third_login = user_time.iloc[i+2]['time_stamp']

                #Check if the time of third login is in 7 days window
                if window >= third_login:
                    s+=1
                    if s!=0:
                        adopted = 1
                    else:
                        adopted = 0

        data = {'user_id': j, 'is_adopted': adopted}
        df = df.append(data, ignore_index=True)

    return df
```

I engineered several features to improve our model's predictive power:

- Extracted date-related features from the user data, such as creation year, month, day, and day of the week.
- Calculated the time since account creation in days.
- Created dummy variables for the 'creation_source' categorical feature.

The engagement data was aggregated at the user level. We then merged both datasets based on the 'user_id' to create a consolidated dataset for modeling.

I chose the Random Forest Classifier for prediction. The data was split into training and testing sets. The model was trained, and its performance was evaluated using accuracy and a classification report.

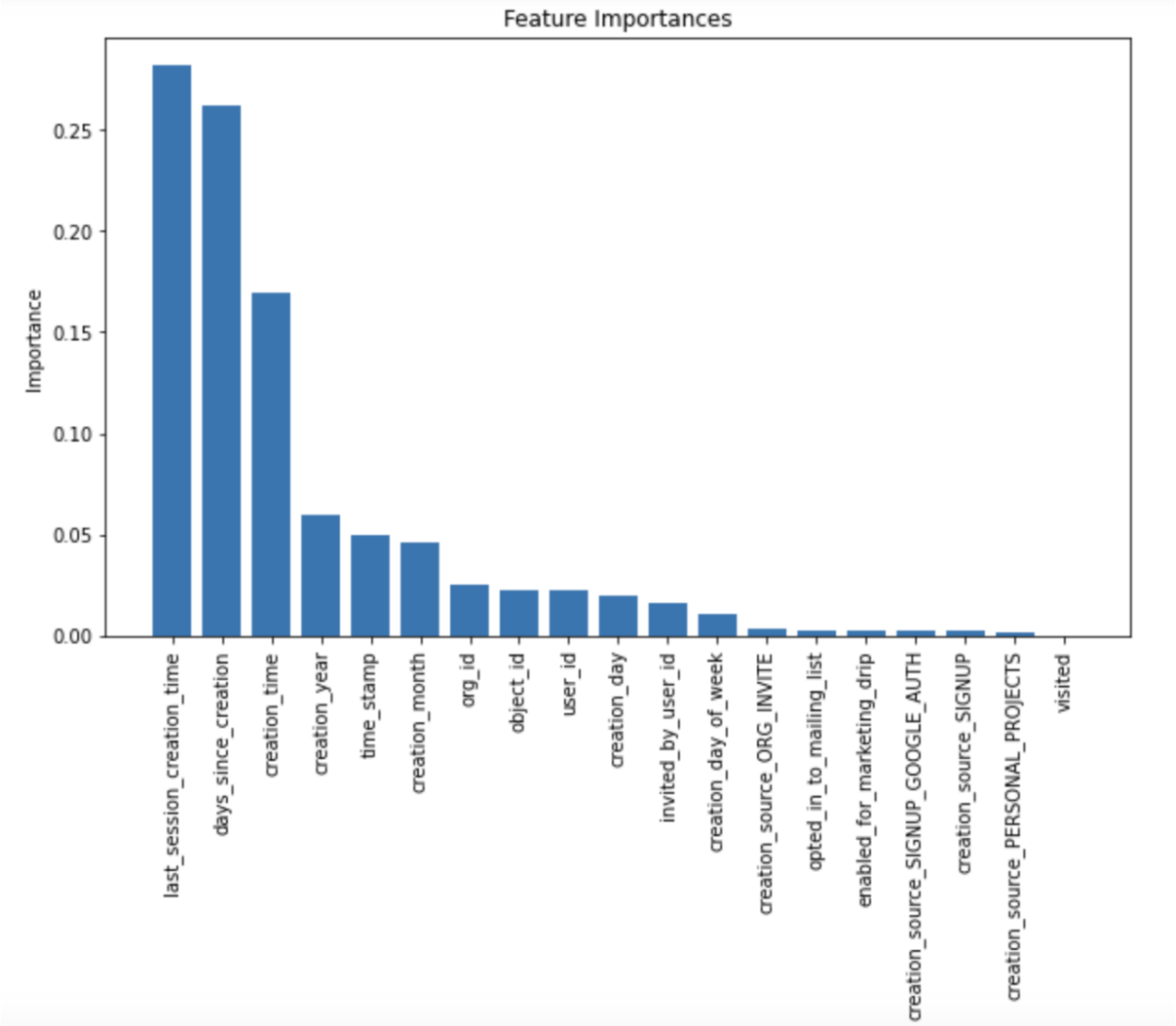
Accuracy: 1.00

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 1.00 | 1.00 | 1773 |
| 1 | 1.00 | 1.00 | 1.00 | 39811 |
| accuracy | | | 1.00 | 41584 |
| macro avg | 1.00 | 1.00 | 1.00 | 41584 |
| weighted avg | 1.00 | 1.00 | 1.00 | 41584 |

Key Findings:

- The Random Forest Classifier achieved an accuracy of 1.00, indicating its ability to predict user adoption.
- Feature Importance Analysis highlighted that last_session_creation_time, days_since_creation, creation_time, creation_year, time_stamp, creation_month are the most influential factors in predicting user adoption.




This analysis opens doors to further research like investigating the impact of factors not included in the current analysis, such as user behavior or product features, on adoption.