Below is a template for Asana's early career data dcience take-home assessment. Although we encourage candidates to use a similar format as below, feel free to make changes as needed!

Data Ingestion

```
# How to read the data files in Python

import pandas as pd
%pip install matplotlib
import matplotlib.pyplot as plt
!pip install -U imbalanced-learn
from imblearn.over_sampling import SMOTE
%pip install sklearn
import sklearn
from sklearn import preprocessing, metrics
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import linear_model
%pip install numpy
import numpy as np
import statsmodels.api as sm
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from xgboost import XGBClassifier
from numpy import mean
from numpy import std
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: matplotlib in /usr/local/lib/python3.8/dist-packages (3.2.2)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib) (1.4.4)
```

```
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.8/dist-packages (from matplotlib) (1.21.6)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.8/dist-packages (from
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.1->matplotli
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.8/dist-packages (0.9.1)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.8/dist-packages (from imbalanced-learn) (1.21.6)
Requirement already satisfied: scikit-learn>=1.1.0 in /usr/local/lib/python3.8/dist-packages (from imbalanced-learn) (1
Requirement already satisfied: joblib>=1.0.0 in /usr/local/lib/python3.8/dist-packages (from imbalanced-learn) (1.2.0)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.8/dist-packages (from imbalanced-learn) (1.7.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from imbalanced-learn) (
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: sklearn in /usr/local/lib/python3.8/dist-packages (0.0.post1)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (1.21.6)
```

```
users = pd.read_csv("https://s3.amazonaws.com/asana-data-interview/takehome_users-intern.csv")
user_engagement = pd.read_csv("https://s3.amazonaws.com/asana-data-interview/takehome_user_engagement-intern.csv")
```

1) Calculating Adoption Rate

```
counts_of_user = user_engagement.groupby("user_id")["user_id"].transform(len)
mask = (counts_of_user > 2)
user_engagement = user_engagement[mask]
user_engagement.time_stamp = pd.to_datetime(user_engagement.time_stamp)
users.rename(columns={'object_id':'user_id'}, inplace=True)
users.head()
```

	user_id	creation_time	name	email	creation_source	last_session_creation_time	opted_in_to
0	1	4/22/14 3:53	Clausen August	AugustCClausen@yahoo.com	GUEST_INVITE	1.398139e+09	

```
adopted_dict = {x: False for x in range(1, len(users)+1)}
from datetime import datetime
adoption_count = 0
grouped_users = user_engagement.groupby('user_id')  # grouping the main user database

df = user_engagement['time_stamp'].sort_values().reset_index(drop = True)
for group in grouped_users:
  user_id = group[0]

  for i, timestamp in enumerate(df):
    if i == len(group[1]['time_stamp']) - 2:  # making sure that the dates selected have 2 dates after it
      break
    start_time = timestamp
    end_time = start_time + pd.Timedelta('7D')  #adding 7 days to the end date
    time_1 = df[i+1]
    time_2 = df[i+2]
    if (time_1 < end_time) and (time_2 < end_time):  #checcking the times of the 2 consecutives dates after selected one
      adoption_count += 1
      adopted_dict[user_id] = True  # creating a dictionary key for each user id and if they are adopted then store value as
      #print(adopted_dict)
      break


  #df.iloc by each userid
  #if count is greater than 3:
  #add to adopted user count


#print(adopted_dict)
adopted_users = pd.DataFrame(list(adopted_dict.items()), columns=['user_id', 'adopted'])
users = pd.merge(users, adopted_users, on='user_id', how='outer')
```

```
{1: False, 2: True, 3: False, 4: False, 5: False, 6: False, 7: False, 8: False, 9: False, 10: True, 11: False, 12: Fals
```

```
amount_of_users = len(users['user_id'])
count = 0
for user in adopted_dict.values():
  if user == True:
    count += 1
print("Amount of Adopted Users: " + str(count))
print("Amount of total users:" + str(amount_of_users))
print(count/amount_of_users)
#print(users)
```

```
Amount of Adopted Users: 2248
Amount of total users:12000
0.18733333333333332
       user_id  creation_time               name                       email  \
0            1   4/22/14 3:53     Clausen August    AugustCClausen@yahoo.com
1            2  11/15/13 3:45      Poole Matthew      MatthewPoole@gustr.com
2            3  3/19/13 23:14  Bottrill Mitchell  MitchellBottrill@gustr.com
3            4   5/21/13 8:09    Clausen Nicklas   NicklasSClausen@yahoo.com
4            5  1/17/13 10:14         Raw Grace          GraceRaw@yahoo.com
...        ...            ...                ...                         ...
11995    11996    9/6/13 6:14       Meier Sophia       SophiaMeier@gustr.com
11996    11997  1/10/13 18:28      Fisher Amelie      AmelieFisher@gmail.com
11997    11998  4/27/14 12:45        Haynes Jake         JakeHaynes@cuvox.de
11998    11999  5/31/12 11:55       Faber Annett         mhaerzxp@iuxiw.com
11999    12000   1/26/14 8:57         Lima Thaìs  ThaisMeloLima@hotmail.com

          creation_source  last_session_creation_time  \
0            GUEST_INVITE                1.398139e+09
1              ORG_INVITE                1.396238e+09
2              ORG_INVITE                1.363735e+09
3            GUEST_INVITE                1.369210e+09
4            GUEST_INVITE                1.358850e+09
...                   ...                         ...
11995          ORG_INVITE                1.378448e+09
11996  SIGNUP_GOOGLE_AUTH                1.358275e+09
11997        GUEST_INVITE                1.398603e+09
11998   PERSONAL_PROJECTS                1.338638e+09
11999              SIGNUP                1.390727e+09

       opted_in_to_mailing_list  enabled_for_marketing_drip  org_id  \
0                             1                           0      11
```

```
1                             0                            0       1
2                             0                            0      94
3                             0                            0       1
4                             0                            0     193
...                         ...                          ...     ...
11995                         0                            0      89
11996                         0                            0     200
11997                         1                            1      83
11998                         0                            0       6
11999                         0                            1       0

       invited_by_user_id email_domain  adopted
0                 10803.0    yahoo.com    False
1                   316.0    gustr.com     True
2                  1525.0    gustr.com    False
3                  5151.0    yahoo.com    False
4                  5240.0    yahoo.com    False
...                   ...          ...      ...
11995              8263.0    gustr.com    False
11996                 NaN    gmail.com    False
11997              8074.0     cuvox.de    False
11998                 NaN    iuxiw.com    False
11999                 NaN  hotmail.com    False

[12000 rows x 12 columns]
```

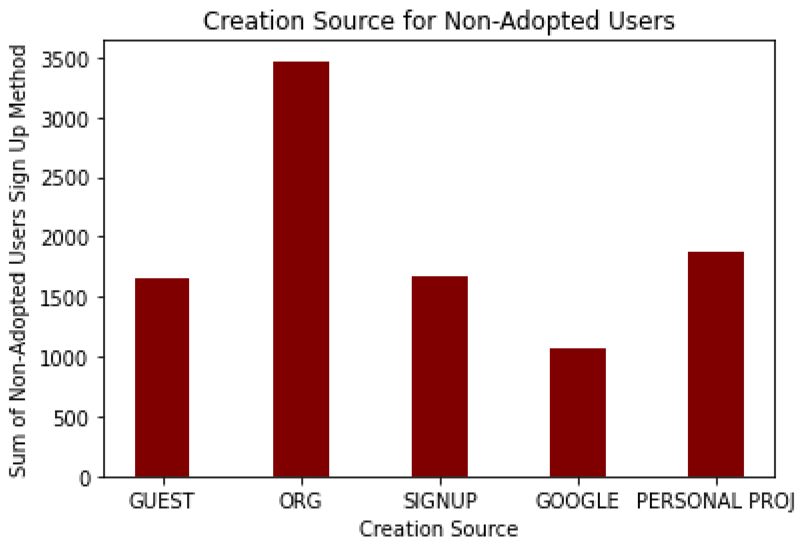
The adoption rate is approximately 18.7%. I found the adoption rate by filtering the users who have more than 3 logins. After that, I created a grouping of user to find out whether they log in more than 3 times.

2) Methodology

```
fig = plt.figure()
x1 = ['GUEST', 'ORG', 'SIGNUP', 'GOOGLE', 'PERSONAL PROJ']
y1 = [(users[users["adopted"] == False]["creation_source"] == 'GUEST_INVITE').sum(), #adding up all values of the creation ty
     (users[users["adopted"] == False]["creation_source"] == 'ORG_INVITE').sum(),
     (users[users["adopted"] == False]["creation_source"] == 'SIGNUP').sum(),
```

```
     (users[users["adopted"] == False]["creation_source"] == 'SIGNUP_GOOGLE_AUTH').sum(),
     (users[users["adopted"] == False]["creation_source"] == "PERSONAL_PROJECTS").sum()]
data1 = dict(zip(x1, y1))
plt.bar(list(data1.keys()), list(data1.values()), color='maroon', width = 0.4)
plt.title('Creation Source for Non-Adopted Users')
plt.xlabel('Creation Source')
plt.ylabel('Sum of Non-Adopted Users Sign Up Method')
plt.show()

fig = plt.figure()
x2 = ['GUEST', 'ORG', 'SIGNUP', 'GOOGLE', 'PERSONAL PROJ']
y2 = [(users[users["adopted"] == True]["creation_source"] == 'GUEST_INVITE').sum(),
     (users[users["adopted"] == True]["creation_source"] == 'ORG_INVITE').sum(),
     (users[users["adopted"] == True]["creation_source"] == 'SIGNUP').sum(),
     (users[users["adopted"] == True]["creation_source"] == 'SIGNUP_GOOGLE_AUTH').sum(),
     (users[users["adopted"] == True]["creation_source"] == "PERSONAL_PROJECTS").sum()]
data2 = dict(zip(x2, y2))
plt.bar(list(data2.keys()), list(data2.values()), color='maroon', width = 0.4)
plt.title('Creation Source for Adopted Users')
plt.xlabel('Creation Source')
plt.ylabel('Sum of Adopted Users Sign Up Method')
plt.show()
```



From the bar graphs above, it is clear that the organization invites are the most popular type of creation for both non-adopted users and adopted users. Now that we know that, I want to move on to figuring out if there are other factors affecting adopted user rate and also to figure out whether organization signups are statistically significant.

```
creation_source_dummy_var = pd.get_dummies(users['creation_source'], prefix='creation_source') #adding dummy variables for ea
creation_source_dummy_var.head()
weekday_df = users['creation_time'].apply(lambda x : pd.to_datetime(x).weekday())  # creating new feature called creation tin
users['weekday'] = weekday_df.apply(lambda x: 1 if x < 5 else 0)  # it will be a dummy variable, thus it is measured in 0 and
#users['weekday'].head()
```

```
0    1
1    1
2    1
3    1
4    1
Name: weekday, dtype: int64
```

```
users = users.join(creation_source_dummy_var)
#users.describe()
```

	user_id	last_session_creation_time	opted_in_to_mailing_list	enabled_for_marketing_drip	org_id	inv
count	12000.00000	8.823000e+03	12000.000000	12000.000000	12000.000000	
mean	6000.50000	1.379279e+09	0.249500	0.149333	141.884583	
std	3464.24595	1.953116e+07	0.432742	0.356432	124.056723	
min	1.00000	1.338452e+09	0.000000	0.000000	0.000000	
25%	3000.75000	1.363195e+09	0.000000	0.000000	29.000000	
50%	6000.50000	1.382888e+09	0.000000	0.000000	108.000000	
75%	9000.25000	1.398443e+09	0.000000	0.000000	238.250000	
max	12000.00000	1.402067e+09	1.000000	1.000000	416.000000	

```
import math
users['invited'] = users['invited_by_user_id'].apply(lambda x: 0 if math.isnan(x) else 1)  # new feature to figure out if use
users.creation_time = pd.to_datetime(users.creation_time)
users.last_session_creation_time = pd.to_datetime(users.last_session_creation_time, unit='s')
users['creation_month'] = users.creation_time.dt.month  # creation month feature to figure out month of creation
users['last_session_year'] = users.last_session_creation_time.dt.year  # new feature to figure out year the user last logged
users.last_session_year.fillna(0, inplace=True)
#print(users['last_session_year'])


predict = "adopted"
ind_vars = ['weekday', 'invited', 'creation_month', 'last_session_year', 'opted_in_to_mailing_list', 'enabled_for_marketing_
X = users[ind_vars]
y = users[predict]
X_train, X_test, y_train, y_test = sklearn.model_selection.train_test_split(X, y, test_size = 0.1, random_state = 0)
linear = linear_model.LinearRegression()
linear.fit(X_train, y_train)
acc = linear.score(X_test, y_test)
print(acc)
```

```
0.09916431979380824
```

```
logit = sm.Logit(y, X)
result = logit.fit()
result.summary()
```

```
Optimization terminated successfully.
```

The accuracy of this regression model and r^2 is very low meaning the results may not be as strong as indicated. I am performing logistic regression to figure out the strength of the relation of between all of the potential factors of adoption(listed on the table above) and adoption. In order to improve the R^2 value and the logit regression, I will use the statistical SMOTE technique to try to make the data more balanced. SMOTE will help balance the distribution of the data randomly to help improve the results.

```
oversampling = SMOTE(random_state=0)  # smote of x and y data
X, y = oversampling.fit_resample(X, y)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
os_data_X, os_data_y = oversampling.fit_resample(X_train, y_train)
os_data_X = pd.DataFrame(data=os_data_X, columns=X_train.columns)
os_data_y = pd.DataFrame(data=os_data_y, columns=[predict])
#print(len(os_data_X))
#print(len(os_data_y))
```

```
13656
13656
```

2a) Writeup associated with methodology

I am using logistic regression to figure out if there is a relationship between the different features and adoption. Adoption is a dummy variable(measured in 0 and 1) to indicate if the user is an adopted user. In order to get the data ready, I cleaned and organized the data using Python Pandas. The modeling method I used was descriptive modeling as I want to find out if a user's creation source or any other factors is likely to increase the change of a user becoming an adopted user. As a result, I added more attributes that could indicate the user's preference. They are listed below:

1. Weekday: dummy variable which indicates if user created account on a weekday
2. Invited: dummy variable which indicates if user was invited by another current Asana user
3. creation_month: the month when user created the account
4. Last_session_year: the year user logged into the account

3) What Factors Predict User Adoption?

```
X = os_data_X.drop([], axis=1)
y = os_data_y
model = XGBClassifier()
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=0)
#print(X.shape)
#print(y.shape)
n_scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
print('Accuracy: %.3f (%.3f)' % (mean(n_scores), std(n_scores)))
xgb_model=model.fit(X,y)
xgb_fea_imp=pd.DataFrame(list(xgb_model.get_booster().get_fscore().items()),
columns=['feature','importance']).sort_values('importance', ascending=False)  # feature of xgb models to figure out importan
print('',xgb_fea_imp)
```

```
Accuracy: 0.797 (0.011)
/usr/local/lib/python3.8/dist-packages/sklearn/preprocessing/_label.py:98: DataConversionWarning: A column-vector y was
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.8/dist-packages/sklearn/preprocessing/_label.py:133: DataConversionWarning: A column-vector y wa
  y = column_or_1d(y, warn=True)
                            feature  importance
0                 last_session_year         193
1                    creation_month         162
3         enabled_for_marketing_drip         23
2        creation_source_GUEST_INVITE        20
4            opted_in_to_mailing_list        18
7                             weekday        17
9              creation_source_SIGNUP        17
8  creation_source_SIGNUP_GOOGLE_AUTH        15
6   creation_source_PERSONAL_PROJECTS        11
5                             invited         9
```

```
X = os_data_X
y = os_data_y
logit = sm.Logit(y, X)
```

```
result = logit.fit()
result.summary()
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.474921
         Iterations: 35
/usr/local/lib/python3.8/dist-packages/statsmodels/discrete/discrete_model.py:1810: RuntimeWarning: overflow encountere
  return 1/(1+np.exp(-X))
/usr/local/lib/python3.8/dist-packages/statsmodels/base/model.py:566: ConvergenceWarning: Maximum Likelihood optimizati
  warnings.warn("Maximum Likelihood optimization failed to "
/usr/local/lib/python3.8/dist-packages/statsmodels/discrete/discrete_model.py:1810: RuntimeWarning: overflow encountere
  return 1/(1+np.exp(-X))
```

Logit Regression Results

Dep. Variable:	adopted	**No. Observations:**	13656
Model:	Logit	**Df Residuals:**	13645
Method:	MLE	**Df Model:**	10
Date:	Mon, 05 Dec 2022	**Pseudo R-squ.:**	0.3148
Time:	02:28:16	**Log-Likelihood:**	-6485.5
converged:	False	**LL-Null:**	-9465.6
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
weekday	-0.1421	0.047	-2.994	0.003	-0.235	-0.049
invited	-3037.4609	1.53e+06	-0.002	0.998	-3e+06	3e+06
creation_month	0.1155	0.007	17.150	0.000	0.102	0.129
last_session_year	1.5256	0.036	42.962	0.000	1.456	1.595
opted_in_to_mailing_list	-0.2190	0.059	-3.717	0.000	-0.334	-0.104
enabled_for_marketing_drip	-0.3078	0.072	-4.257	0.000	-0.449	-0.166
creation_source_GUEST_INVITE	-34.0204	1.53e+06	-2.22e-05	1.000	-3e+06	3e+06
creation_source_ORG_INVITE	-34.4099	1.53e+06	-2.25e-05	1.000	-3e+06	3e+06
creation_source_PERSONAL_PROJECTS	-3071.6851	71.514	-42.952	0.000	-3211.850	-2931.520
creation_source_SIGNUP	-3071.9981	71.515	-42.956	0.000	-3212.165	-2931.831
creation_source_SIGNUP_GOOGLE_AUTH	-3072.0769	71.516	-42.956	0.000	-3212.246	-2931.908

Possibly complete quasi-separation: A fraction 0.16 of observations can be perfectly predicted. This might indicate that there is complete quasi-separation. In this case some parameters will not be identified.

3a) Writeup associated with what factors predict user adoption?

The results from the regression model show that there are 8 important factors that could help predict user adoption. They are weekday, creation_month, last_session_year, opted_in_to_mailing_list, enabled_for_marketing_drip, creation_source_PERSONAL_PROJECTS, creation_source_SIGNUP, and creation_source_SIGNUP_GOOGLE_AUTH. All the p-values of this model are below 0.05 which make them statistically significant. However, with there being so many indicative factors, I also wanted to incorporate the Extreme Gradient Booster model which indicates which features are important. This will help narrow down the important features. As the results from the model show, last_session_year and creation_month far outweigh the rest of the features in terms of importance for whether a user will become an adopted user or not. The coefficients of the regression also indicate the following:

1. Customers who have been active as of late have a higher chance of being an adopted user. It has the highest coefficient out of the rest meaning that for every increase in last_session_year, the odds of being an adopted user go up.
2. Creation month also has a slight positive impact on the chances of becoming an adopted user. I would recommend the Asana team to look deeper into the months that are popular and maybe create promotions around the months that are more popular. The creation_month being a strong indicative factor of adoption rate has huge implications for product strategy as it can help tailor advertisement as well as customer retention in certain seasons or months.
3. Despite the fact that organization signups are so popular, the coeeficient and p-value for them are extremely negative coefficient and statistically insignificant, which means it is not effective in indicating adoption.
4. While some of the creation sources are significant, the negative coefficients as well as the low feature importance score is telling as to the importance of creation sources. The most important creation source seems to be guest invite as per the Gradient Boost model.

My overall takeway is that the last_session_year and creation_month is extremely important in determining adoption.

4) Additional Commentary (Optional)

With more time, I would have liked to explore the correlation between some of the variables in the data provided to eliminate any bias in my methodology. From the initial regression tables, it seems as if the SMOTE method changed some of the coefficients and made them more negative. I would have liked to explore why that is the case and refined my model more. Additionally, I would have liked to look deeper into the creation_month as it became an unlikely feature with increased importance. I would suggest a deeper dive into that feature since it could have impact effects on the product strategy and aim into Asana product and advertising.

```
# This is formatted as code
```