**Analysis of Github User Behavior’s Impact on Their Contribution**

Yanwu Ren (1589701)

Zichen Tian (1596213)

Github Link: <https://github.com/renyanwu/720project>

Step 1 Downloaded Dataset:

We downloaded the raw data from Google BigQuery. We wrote the following SQL stataments to got the data from Jan 1, 2018 to Dec 31, 2018 and we used the 'count' and 'group by' key words to get the times of each event of each user in each month. Due to the extraction limition in Google BigQuery, we had to download the dataset month by month. There are totally 475,516,715 events in 14 categories.

--------------------------------------------------------------

SELECT actor.id, type, count(\*) as n FROM [githubarchive:month.201812] group by actor.id, type

--------------------------------------------------------------

Step 2 Make the feature dataset:

When got the dataset, we merged the event data of a user together and then merge the data of each month together. Finally, the dimension of the dataset is 29,205,739 \* 14. The Python codes we used are showed below:

--------------------------------------------------------------

import pandas as pd

for i in range(1, 13):

if i < 10:

inname = "data/20180" + str(i)

outname = "data/2018\_0" + str(i) + ".csv"

else:

inname = "data/2018" + str(i)

outname = "data/2018\_" + str(i) + ".csv"

df = pd.read\_csv(inname)

default\_dict = {'DeleteEvent': 0, 'PullRequestReviewCommentEvent': 0, 'WatchEvent': 0, 'CommitCommentEvent': 0,

'ReleaseEvent': 0, 'GollumEvent': 0, 'IssueCommentEvent': 0, 'ForkEvent': 0, 'PushEvent': 0, 'CreateEvent': 0,

'IssuesEvent': 0, 'MemberEvent': 0, 'PullRequestEvent': 0, 'PublicEvent': 0}

users = {}

for i, row in df.iterrows():

if row['actor\_id'] not in users:

users[row['actor\_id']] = default\_dict.copy()

users[row['actor\_id']][row['type']] = row['n']

out = pd.DataFrame(users.values())

print(out.head())

out.to\_csv(outname, index = False)

import pandas as pd

df = pd.read\_csv('data/2018\_01.csv')

for i in range(2, 10):

df\_new = pd.read\_csv('data/2018\_0' + str(i) + '.csv')

df = pd.concat([df, df\_new])

for i in range(10, 13):

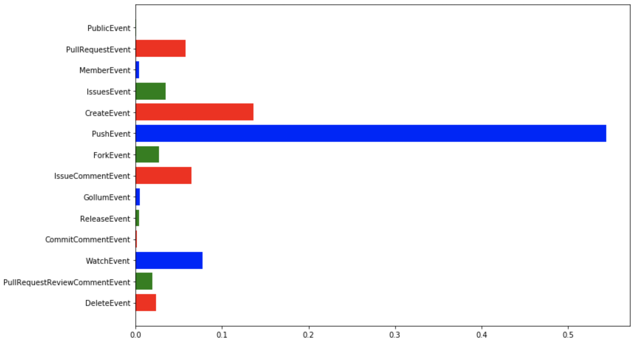
df = pd.read\_csv('data/2018\_' + str(i) + '.csv')

df = pd.concat([df, df\_new])

df.to\_csv('data/total.csv', index = False)

--------------------------------------------------------------

Step 3 Draw the distribution:



--------------------------------------------------------------

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv('data/total.csv')

print(df.columns.values)

arr = []

for col in df.columns.values:

arr.append(sum(df[col]))

arr = [float(x) / sum(arr) for x in arr]

plt.figure(figsize=(12,8))

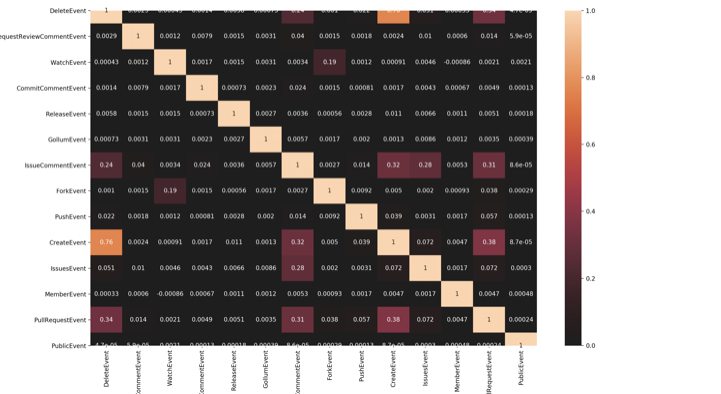
plt.barh(range(len(arr)), arr,color='rgb',tick\_label=df.columns.values)

plt.show()

--------------------------------------------------------------

Step 4 Pearson correlation coefficient:

We calculate the pearson correlation coefficient on the whole dataset and draw the graph:



--------------------------------------------------------------

import pandas as pd

import matplotlib.pyplot as mp

import seaborn

df = pd.read\_csv("data/total.csv")

print(df.head())

df\_corr = df.corr()

seaborn.heatmap(df\_corr, center=0, annot=True)

mp.show()

--------------------------------------------------------------

Step 5: Calculate the interaction value and contribution value:

We defined interaction value and contribution value and calculate the values for each user. After we draw the trend graph during 12 months.

--------------------------------------------------------------

import pandas as pd

import matplotlib.pyplot as plt

act = ['WatchEvent', 'IssueCommentEvent','IssuesEvent', 'MemberEvent','PullRequestEvent','ForkEvent']

cont = ['CreateEvent', 'DeleteEvent', 'PushEvent', 'CommitCommentEvent', 'ReleaseEvent','PublicEvent', 'PullRequestReviewCommentEvent', 'GollumEvent']

sm = []

sact = []

scont = []

for i in range(1, 10):

df = pd.read\_csv('data/2018\_0' + str(i) + '.csv')

sm.append(sum([sum(df[col]) for col in df.columns.values]))

sact.append(sum([sum(df[col]) for col in act]))

scont.append(sum([sum(df[col]) for col in cont]) - 2 \* sum(df['DeleteEvent']))

for i in range(10, 13):

df = pd.read\_csv('data/2018\_' + str(i) + '.csv')

sm.append(sum([sum(df[col]) for col in df.columns.values]))

sact.append(sum([sum(df[col]) for col in act]))

scont.append(sum([sum(df[col]) for col in cont]) - 2 \* sum(df['DeleteEvent']))

print(sm)

print(sact)

print(scont)

month = list(range(1, 13))

plt.figure(figsize=(12,8))

l1=plt.plot(month,sm,'r--',label='number of events')

l2=plt.plot(month,sact,'g--',label='activities')

l3=plt.plot(month,scont,'b--',label='contributions')

plt.plot(month,sm,'ro-',month,sact,'g+-',month,scont,'b^-')

plt.title('Events in different months')

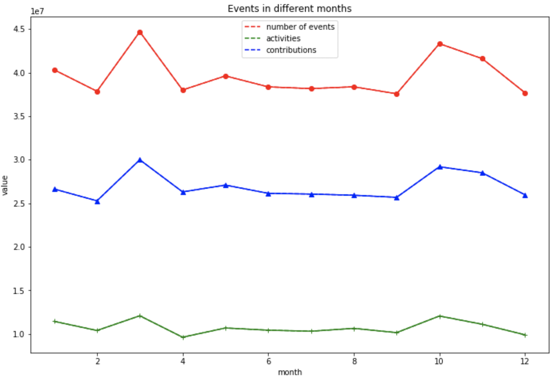
plt.xlabel('month')

plt.ylabel('value')

plt.legend()

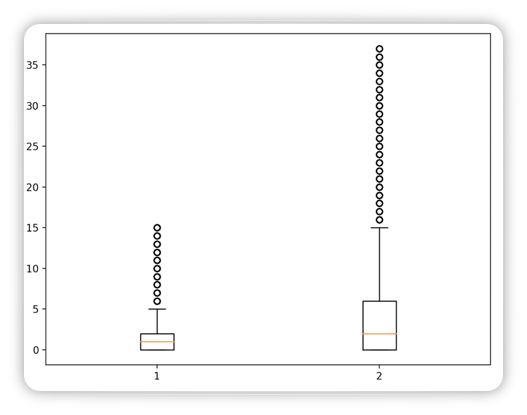
plt.show()

--------------------------------------------------------------



Step 6 Remove extreme values:

After saw the distribution of interaction and contribution, there are many extreme high values and we use 98% quantile threshold to make the data in a reasonable range. However, after saw the distribution of interaction and contribution, there are not a clear relation between them. Therefore, we need to use the use the separate interaction data as the features.



--------------------------------------------------------------

import pandas as pd

import matplotlib.pyplot as plt

import seaborn

import xgboost as xgb

from sklearn.model\_selection import train\_test\_split

import numpy as np

df = pd.read\_csv('total.csv')

cont = []

for i, row in df.iterrows():

cont.append(row['CreateEvent'] - row['DeleteEvent'] + row['PushEvent'] + row['CommitCommentEvent'] \

+ row['ReleaseEvent'] + row['PublicEvent'] + row['PullRequestReviewCommentEvent'])

contp = np.percentile(cont, 98)

df = df[['WatchEvent', 'IssueCommentEvent', 'IssuesEvent', 'MemberEvent', 'PullRequestEvent', 'ForkEvent']]

df['contribution'] = cont

df = df[(df['contribution'] <= contp) & (df['contribution'] >= 0)]

X = df.values[:,:-1]

y = df.values[:, -1]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

dtrain = xgb.DMatrix(X\_train, y\_train)

dtest = xgb.DMatrix(X\_test, y\_test)

--------------------------------------------------------------

Step 7 Train XGBoost Model and tune parameters:

We used 20 % for testing and 80% for training and used cross-validation to get the best parameter combinations:

--------------------------------------------------------------

import pandas as pd

import matplotlib.pyplot as plt

import seaborn

import xgboost as xgb

from sklearn.model\_selection import train\_test\_split

import numpy as np

from sklearn.model\_selection import GridSearchCV

def tune():

cv\_params = {'n\_estimators': [500, 1000, 1500, 2000, 2500], 'learning\_rate': [0.001, 0.01, 0.1, 0.5, 1],

'max\_depth': [3, 4, 5, 6], 'min\_child\_weight': [1, 2, 3]}

other\_params = {'learning\_rate': 0.1, 'n\_estimators': 500, 'max\_depth': 5, 'min\_child\_weight': 1, 'seed': 0,

'subsample': 0.8, 'colsample\_bytree': 0.8, 'gamma': 0, 'reg\_alpha': 0, 'reg\_lambda': 1}

model = xgb.XGBRegressor(\*\*other\_params)

optimized\_GBM = GridSearchCV(estimator=model, param\_grid=cv\_params, scoring='r2', cv=5, verbose=1, n\_jobs=4)

return optimized\_GBM

optimized\_GBM = tune()

optimized\_GBM.fit(X\_train, y\_train)

print('best\_param：{0}'.format(optimized\_GBM.best\_params\_))

print('best\_sccore:{0}'.format(optimized\_GBM.best\_score\_))

--------------------------------------------------------------

Step 8 Results and Importance:

We used the best parameters to train the XGBoost model to get the best parameters and the importance of features.

--------------------------------------------------------------

other\_params = {'learning\_rate': 0.1, 'n\_estimators': 2000, 'max\_depth': 5, 'min\_child\_weight': 1, 'seed': 0,

'subsample': 0.8, 'colsample\_bytree': 0.8, 'gamma': 0, 'reg\_alpha': 0, 'reg\_lambda': 1}

model = xgb.XGBRegressor(\*\*other\_params)

model.fit(X\_train, y\_train)

ans = model.predict(X\_test)

error = [abs(ans[i] - y\_test[i]) for i in range(len(ans))]

print(np.mean(error))

print([error[i] / y\_test[i] for i in range(len(ans))])

plt.figure(figsize=(12,6))

plt.barh(range(len(model.feature\_importances\_)), model.feature\_importances\_, tick\_label = ['WatchEvent', 'IssueCommentEvent', 'IssuesEvent', 'MemberEvent', 'PullRequestEvent', 'ForkEvent'])

plt.show()

--------------------------------------------------------------

