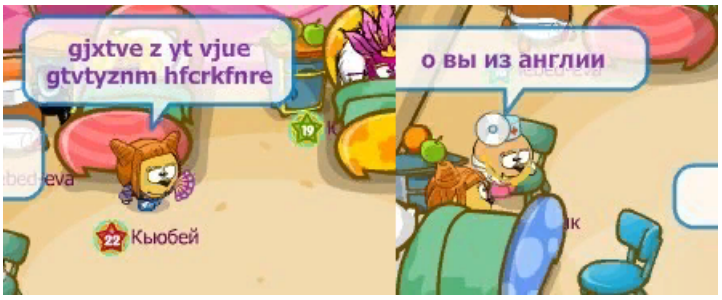
Отчёт по ЛР №2 по ТМО

Водка Игорь, ИУ5-61



Example of using pandasql library for data analysis

In [ ]:
```
%matplotlib inline
import pandas as pd
import pandasql as ps
from datetime import datetime
import seaborn
import matplotlib.pyplot as plt

%config InlineBackend.figure_format = 'svg'
from pylab import rcParams
rcParams['figure.figsize'] = 8, 5
```

In [ ]:
```
pd.__version__
```

In [ ]:
```
project_submissions = pd.read_csv('./data/project_submissions.csv')
daily_engagements = pd.read_csv('./data/daily_engagement.csv')
enrollments = pd.read_csv('./data/enrollments.csv')
```

Simple SQL query

getting accounts and date with maximum total time spent on Udacity

In [ ]:
```
# pandasql code
def example1_pandasql(daily_engagements):
    simple_query = '''
        SELECT
            acct,
            total_minutes_visited,
            utc_date
        FROM daily_engagements
        ORDER BY total_minutes_visited desc
        LIMIT 10
        '''
    return ps.sqldf(simple_query, locals())

# pandas code
def example1_pandas(daily_engagements):
    return daily_engagements[['acct', 'total_minutes_visited', 'utc_date']].sort_values(
by ='total_minutes_visited', ascending = False)[:10]
```

In [ ]:
```
example1_pandasql(daily_engagements)
```

In [ ]:
```
example1_pandas(daily_engagements)
```

SQL query with aggregating functions

Let's see whether there's weekly seasonality: on average students spent more time on weekends then on weekdays

In [ ]:
```
# ТУТ НЕ РАБОТАЛО. ДОБАВИЛ list() ВОКРУГ map()

daily_engagements['weekday'] = list(map(lambda x: datetime.strptime(x, '%Y-%m-%d').strft
ime('%A'), daily_engagements.utc_date))
```

In [ ]:
```
daily_engagements.head()
```

In [ ]:
```
# pandasql code
def example2_pandasql(daily_engagements):
    aggr_query = '''
        SELECT
            avg(total_minutes_visited) as total_minutes_visited,
            weekday
        FROM daily_engagements
        GROUP BY weekday
        '''
    return ps.sqldf(aggr_query, locals()).set_index('weekday')

# pandas code
def example2_pandas(daily_engagements):
    return pd.DataFrame(daily_engagements.groupby('weekday').total_minutes_visited.mean
())
```

In [ ]:
```
weekday_engagement = example2_pandasql(daily_engagements)
weekday_engagement
```

In [ ]:
```
example2_pandas(daily_engagements)
```

In [ ]:
```
week_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunda
y']
weekday_engagement.loc[week_order].plot(kind = 'bar', rot = 45, title = 'Total time spen
t on Udacity by weekday')
```

Joining tables

Let's see whether students that canceled program was spending less time on Udacity within first week of enrollment. Note we need to filter out Udacity test users not to spoil statistics. Also we need to take into account the fact that student may join several times.

In [ ]:
```
# pandasql code
def example3_pandasql(enrollments, daily_engagements):
    join_query = '''
        SELECT
            avg(avg_acct_total_minutes) as avg_total_minutes,
            status
        FROM
            (SELECT
                avg(total_minutes_visited) as avg_acct_total_minutes,
                status,
                account_key
            FROM
                (SELECT
                    e.account_key,
                    e.status,
                    de.total_minutes_visited,
                    (cast(strftime('%s',de.utc_date) as interger) - cast(strftime('%s',
e.join_date) as interger))/(24*60*60) as days_since_joining,
                    (cast(strftime('%s',e.cancel_date) as interger) - cast(strftime('%s
', de.utc_date) as interger))/(24*60*60) as days_before_cancel
                FROM enrollments as e JOIN daily_engagements as de ON (e.account_key = d
e.acct)
                WHERE (is_udacity = 0) AND (days_since_joining < 7) AND (days_since_join
ing >= 0)
                    AND ((days_before_cancel >= 0) OR (status = 'current'))
                )
            GROUP BY status, account_key)
        GROUP BY status
    '''
    return ps.sqldf(join_query, locals()).set_index('status')

# pandas code
def example3_pandas(enrollments, daily_engagements):
    join_df = pd.merge(daily_engagements,
                    enrollments[enrollments.is_udacity == 0],
                    how = 'inner',
                    right_on ='account_key',
                    left_on = 'acct')
    join_df = join_df[['account_key', 'status', 'total_minutes_visited', 'utc_date', 'jo
in_date', 'cancel_date']]

    join_df['days_since_joining'] = map(lambda x: x.days,
                                        pd.to_datetime(join_df.utc_date) - pd.to_datetim
e(join_df.join_date))

    join_df['before_cancel'] = (pd.to_datetime(join_df.utc_date) <= pd.to_datetime(join_
df.cancel_date))
    join_df = join_df[join_df.before_cancel | (join_df.status == 'current')]

    join_df = join_df[(join_df.days_since_joining < 7) & (join_df.days_since_joining >=
0)]
    avg_account_total_minutes = pd.DataFrame(join_df.groupby(['account_key', 'status'],
as_index = False)
                                                        .total_minutes_visited.mean())
    avg_total_minutes= pd.DataFrame(avg_account_total_minutes.groupby('status').total_mi
nutes_visited.mean())
    avg_total_minutes.columns = ['avg_total_minutes']
    return avg_total_minutes
```

In [ ]:
```
example3_pandasql(enrollments, daily_engagements)
```

In [ ]:
```
example3_pandas(enrollments, daily_engagements)
```

Estimating time elapsed

In [ ]:
```
import time

def count_mean_time(func, params, N =5):
    total_time = 0
    for i in range(N):
        time1 = time.time()
        if len(params) == 1:
            tmp_df = func(params[0])
        elif len(params) == 2:
            tmp_df = func(params[0], params[1])
        time2 = time.time()
        total_time += (time2 - time1)
    return total_time/N
```

Example #1

In [ ]:
```
ex1_times = []
for count in range(1000, 137000, 1000):
    pandasql_time = count_mean_time(example1_pandasql, [daily_engagements[:count]])
    pandas_time = count_mean_time(example1_pandas, [daily_engagements[:count]])
    ex1_times.append({'count': count, 'pandasql_time': pandasql_time, 'pandas_time': pan
das_time})
```

In [ ]:
```
ex1_times_df = pd.DataFrame(ex1_times)
ex1_times_df.columns = ['number of rows in daily_engagements', 'pandas time', 'pandasql
 time']
ex1_times_df = ex1_times_df.set_index('number of rows in daily_engagements')
```

In [ ]:
```
ax = ex1_times_df.plot(title = 'Example #1 time elapsed (seconds)', subplots = True)
```

Example #2

In [ ]:
```
ex2_times = []
for count in range(1000, 137000, 1000):
    pandasql_time = count_mean_time(example2_pandasql, [daily_engagements[:count]])
    pandas_time = count_mean_time(example2_pandas, [daily_engagements[:count]])
    ex2_times.append({'count': count, 'pandasql_time': pandasql_time, 'pandas_time': pan
das_time})
```

In [ ]:
```
ex2_times_df = pd.DataFrame(ex2_times)
```

In [ ]:
```
ex2_times_df.columns = ['number of rows in daily_engagements', 'pandas time', 'pandasql
 time']
ex2_times_df = ex2_times_df.set_index('number of rows in daily_engagements')
```

In [ ]:
```
ax = ex2_times_df.plot(title = 'Example #2 time elapsed (seconds)', subplots = True)
```

Example #3

In [ ]:
```
all_users = enrollments.account_key.unique().tolist()
len(all_users)
```

In [ ]:
```
ex3_times = []
for users_count in range(10, 1310, 10):
    users = all_users[:users_count]
    enrollments_sample = enrollments[enrollments.account_key.isin(users)]
    daily_engagements_sample = daily_engagements[daily_engagements.acct.isin(users)]
    count = daily_engagements_sample.shape[0]
    pandasql_time = count_mean_time(example3_pandasql, [enrollments_sample, daily_engage
ments_sample])
    pandas_time = count_mean_time(example3_pandas, [enrollments_sample, daily_engagement
s_sample])
    ex3_times.append({'count': count, 'pandasql_time': pandasql_time, 'pandas_time': pan
das_time})
```

In [ ]:
```
ex3_times_df = pd.DataFrame(ex3_times).set_index('count')
```

In [ ]:
```
ax = ex3_times_df.plot(title = 'Example #3 time elapsed')
ax.set_xlabel('number of rows in daily_engagements')
ax.set_ylabel('time, seconds')
```

In [ ]:
```

```