Capital Bikeshare: Anlayse und Prognose der Ausleihvorgänge

30 - Ermittlung Anzahl Fahrräder nach Vorgangsart pro Station, Tag und Stunde

Hinweis: Nur Notebooks mit ganzen *10*er-Nummern gehören zur eigentlichen Verarbeitungs-Pipeline und müssen der Größe nach ausgeführt werden, da spätere Notebooks (die mit einer größeren Anfangsnummer) Daten aus den vorherigen Notebooks verwenden.

Relevant sind die Vorgänge Initialisierung (erste Nutzung eines *neuen* Fahrrades), Umverteilungen zwischen Stationen sowie reguläre Entnahmen und Rückgaben sowie die Deltas bezogen auf alle Vorgänge und die kumulierten Summen pro Station.

In [1]:

```
import datetime
import pandas as pd
```

In [2]:

```
DATA_PATH = '../data/'
TRIPS_FILE = 'trips_clean.pkl'
COUNTS_FILE = 'counts.pkl'
COUNT_KEY_LIST = ['station_id', 'date', 'hour', 'Member type']
```

In [ ]:

```

```

In [3]:

```
df_trips = pd.read_pickle(DATA_PATH+TRIPS_FILE)
```

In [4]:

```
df_trips.head()
```

Out[4]:

	start_ts	end_ts	start_station_id	end_station_id	bike_number	Member type
130487	2015-10-15 10:58:35	2015-10-15 14:57:10	31219	31634	? (0x0000000074BEBCE4)	Member
193289	2016-10-18 10:54:16	2016-10-18 11:19:17	31292	31292	? (0x0000000074BEBCE4)	Member
207012	2016-10-19 12:20:37	2016-10-19 12:32:45	31618	31618	? (0x0000000074BEBCE4)	Member
241628	2016-10-22 12:07:42	2016-10-22 12:26:22	31249	31249	? (0x0000000074BEBCE4)	Member
242374	2016-10-22 13:01:26	2016-10-22 13:30:08	31249	31249	? (0x0000000074BEBCE4)	Member

In [ ]:

```

```

In [ ]:

```

```

In [5]:

```
# Funktion zum Gruppieren der Zeilen nach Station, Tag und Stunde
# für Ausleihe (start/out)
# oder Rückgaben (end/in)
def group_by_hour(df, prefix='start', suffix='out'):
    col_station, col_date, col_hour, col_Member = prefix+'_station_id', prefix+'_date',
prefix+'_hour', 'Member type'
    ret =  df.groupby(by=[col_station, col_date, col_hour, 'Member type']
                      )[col_hour].agg('count').to_frame('count_'+suffix)
    # Reihenfolge der COUNT_KEY_LIST muss oben natürlich beachtet werden!
    ret.index.names = COUNT_KEY_LIST
    return ret
```

In [ ]:

```

```

In [6]:

```
# Gruppierung der Leih-Transaktionen für Ausleihen (start/out) und Rückgaben (end/in)
df_count_out = group_by_hour(df_trips, prefix='start', suffix='out')
df_count_in = group_by_hour(df_trips, prefix='end', suffix='in')
```

In [7]:

```
df_count_out.head()
```

Out[7]:

				count_out
station_id	date	hour	Member type	
31000	2015-01-01	10	Member	1
		17	Casual	4
			Member	1
	2015-01-02	11	Member	1
		15	Member	1

In [8]:

```
df_count_in.head()
```

Out[8]:

				count_in
station_id	date	hour	Member type	
31000	2015-01-01	10	Member	1
		23	Member	1
	2015-01-02	16	Member	1
	2015-01-04	15	Member	1
		17	Member	1

In [9]:

```
# Überprüfung der Zahlen ... Summen müssen gleich sein
df_count_out['count_out'].sum(), df_count_in['count_in'].sum()
```

Out[9]:

```
(10277653, 10277653)
```

In [10]:

```
# Ein- und Ausgänge per Outer-Join kombinieren
df_counts = df_count_in.join(df_count_out, on=COUNT_KEY_LIST, how='outer')
```

In [11]:

```
df_counts.head()
```

Out[11]:

				count_in	count_out
station_id	**date**	**hour**	**Member type**		
31000	**2015-01-01**	**10**	**Member**	1.0	1.0
		23	**Member**	1.0	NaN
	2015-01-02	**16**	**Member**	1.0	NaN
	2015-01-04	**15**	**Member**	1.0	NaN
		17	**Member**	1.0	NaN

In [ ]:

```

```

In [12]:

```
# Im Folgenden soll ein vollständiger Index-DataFrame erzeugt werden,
# der für jede Station für jede Stunde zwischen der ersten und letzten
# Entnahme/Rückgabe einen Eintrag hat.
# Der Count-DataFrame wird dann damit per Left-Join verbunden, damit
# der DataFrame lücklos für jede Stunde eine Zeile hat.
```

In [13]:

```
# Erzeuge Index für alle Tage von first bis last mit allen Stunden
def create_hour_index(date_first, date_last):
    date_range = pd.date_range(date_first, date_last)
    hour_range = range(0,24)
    membertype =['Casual', 'Member']
    return pd.MultiIndex.from_product([date_range, hour_range, membertype], names = ['d
ate', 'hour', 'Member type'])
```

In [14]:

```
# Wählte alle Datensätze aus, bei denen Station_ID entsprechend des übergebenen Werts i
st. StationID zählt nicht mehr zu Multi-Index
# In index_first wird kleinste Kombi aus Datum und Stunde gespeichert, in Index_last gr
ößte
def get_df_index_station(df, station_id):
    station_index = df_counts.loc[station_id].index
    index_first = station_index.min()
    index_last = station_index.max()
    df_index = pd.DataFrame(index=create_hour_index(index_first[0], index_last[0]))
    df_index = df_index[index_first:index_last]
    return pd.concat([df_index], keys=[station_id], names=['station_id'])
```

In [15]:

```
# Erzeuge Index-DataFrame mit vollständiger Multi-Level-Index-Struktur
# (Station, Tag, Stunde) für alle Station im Count-DataFrame (df)
# Unterberücksichtigung des ersten und letzten bekannten Eintrags für
# jede Station
def get_df_index(df):
    df_list = [get_df_index_station(df, station_id) for station_id in df.index.levels[0
]]
    return pd.concat(df_list)
```

In [16]:

```
# Index-DataFrame erzeugen
df_index = get_df_index(df_counts)
```

In [17]:

```
# Aktuelle Counts-DataFrame dagegen joinen und überschreiben
df_counts = df_index.join(df_counts, on=COUNT_KEY_LIST, how='left')
```

In [18]:

```
df_counts.head()
```

Out[18]:

				count_in	count_out
station_id	**date**	**hour**	**Member type**		
31000	**2015-01-01**	**10**	**Member**	1.0	1.0
		11	**Casual**	NaN	NaN
			Member	NaN	NaN
		12	**Casual**	NaN	NaN
			Member	NaN	NaN

In [ ]:

```

```

In [19]:

```
# fehlende Werte (keine Trips) auf 0 setzen
df_counts.fillna(value=0.0, inplace=True)
```

In [20]:

```
# Jetzt können alle Count-Werte in integer umgewandelt werden (war float wegen fehlende
r Werte)
df_counts = df_counts.astype(int)
```

In [21]:

```
df_counts.head()
```

Out[21]:

				count_in	count_out
station_id	**date**	**hour**	**Member type**		
31000	**2015-01-01**	**10**	**Member**	1	1
		11	**Casual**	0	0
			Member	0	0
		12	**Casual**	0	0
			Member	0	0

In [ ]:

```

```

In [22]:

```
# Ergebnis speichern
df_counts.to_pickle(DATA_PATH+COUNTS_FILE)
```