In [1]:

```
import pandas as pd
```

In [2]:

```
transactions_train = pd.read_csv('data/transactions_train.csv')

# train_target.csv should be preliminary splitted on target_train.csv and target_te

train_target = pd.read_csv('data/target_train.csv')
test_target = pd.read_csv('data/target_test.csv')
group_desc = pd.read_csv('data/small_group_description.csv')
group_desc_dict = group_desc['small_group'].to_dict()
```

Посмотрим на данные

In [3]:

```
transactions_train.head()
```

Out[3]:

	client_id	trans_date	small_group	amount_rur
0	33172	6	4	71.463
1	33172	6	35	45.017
2	33172	8	11	13.887
3	33172	9	11	15.983
4	33172	10	11	21.341

In [4]:

```
train_target.head()
```

Out[4]:

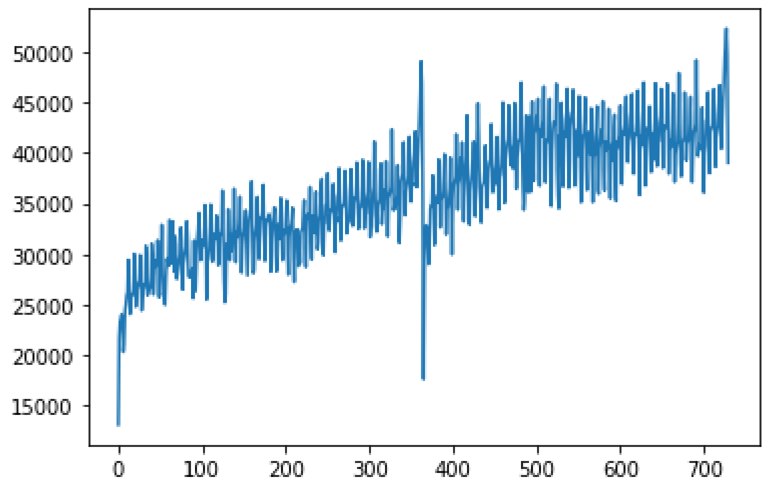
	Unnamed: 0	client_id	bins
0	2308	36627	3
1	22404	16901	1
2	23397	23218	2
3	25058	47465	3
4	2664	18443	0

In [5]:

```
transactions_train['trans_date'].value_counts().sort_index().plot()
```

Out[5]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f823f9f30b8>
```

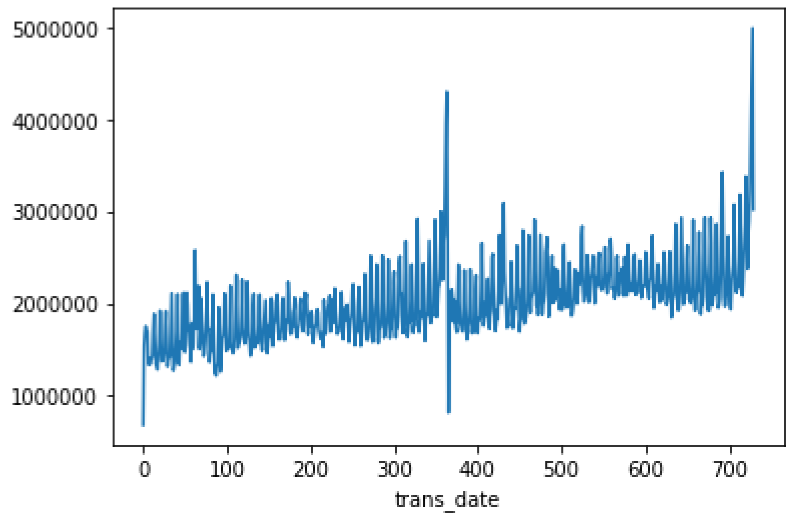


In [6]:

```
transactions_train.groupby('trans_date').sum()['amount_rur'].plot()
```

Out[6]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f823d00c5f8>
```



Отскейлим суммы транзакций

In [7]:

```
transactions_train['amount_scaled'] = (transactions_train['amount_rur']-transaction
```

In [8]:

```
transactions_train.head()
```

Out[8]:

	client_id	trans_date	small_group	amount_rur	amount_scaled
0	33172	6	4	71.463	0.042223
1	33172	6	35	45.017	-0.029352
2	33172	8	11	13.887	-0.113604
3	33172	9	11	15.983	-0.107932
4	33172	10	11	21.341	-0.093430

Объединим данные о транзакциях с таргетом

In [9]:

```
trans_agg = transactions_train.groupby('client_id').agg({'trans_date': list, 'small
df = trans_agg.merge(train_target)
```

In [10]:

```
df.head()
```

Out[10]:

	client_id	trans_date	small_group	amount_scaled	Unnamed: 0	bins
0	4	[0, 2, 3, 4, 6, 6, 8, 9, 9, 10, 10, 11, 11, 12...	[1, 3, 1, 1, 4, 1, 1, 36, 1, 15, 1, 34, 1, 1, ...	[-0.12355881153565301, -0.07620373497423152, -...	23033	1
1	6	[0, 5, 10, 11, 15, 15, 16, 16, 17, 18, 19, 19,...	[15, 3, 1, 3, 15, 1, 15, 3, 37, 11, 3, 1, 3, 1...	[-0.1402171241741367, -0.11400768476047689, -0...	14774	1
2	7	[1, 2, 12, 13, 14, 16, 17, 20, 22, 22, 24, 30,...	[3, 19, 1, 4, 11, 25, 3, 1, 18, 1, 1, 25, 36, ...	[-0.10160935329063224, 0.3758170749878081, -0....	13678	0
3	11	[0, 2, 6, 8, 9, 10, 10, 11, 14, 15, 15, 15, 18...	[3, 25, 1, 1, 11, 25, 1, 1, 1, 3, 1, 11, 22, 2...	[-0.10449986147875583, -0.08921643475746947, -...	16037	3
4	12	[3, 6, 6, 6, 7, 8, 8, 8, 8, 9, 9, 10, 10, 1...	[1, 28, 36, 18, 1, 11, 43, 1, 9, 11, 31, 1, 42...	[-0.14301019950198646, -0.07806307872445707, -...	10495	2

Обучим w2v!

Представим список транзакций для каждого юзера в виде предложений.

1 предложение - 1 неделя

In [11]:

```
from tqdm import tqdm

sents = []

for _, row in tqdm(df.iterrows(), total=len(df)):
    user_sents = []
    dates = row['trans_date']
    groups = row['small_group']
    date_group = {}
    for date, group in zip(dates, groups):
        if date not in date_group:
            date_group[date] = []
        date_group[date].append(str(group))

    for i in range(0, max(date_group.keys()), 7):
        week_sents = []
        for d in range(i, i+7):
            week_sents.extend(date_group.get(d, []))
        sents.append(week_sents)
```

```
100%|██████████| 24000/24000 [00:19<00:00, 1260.68it/s]
```

In [12]:

```
from gensim.models import Word2Vec

wv_model = Word2Vec(sents, size=50, window=5, min_count=1, workers=2)
```

Разделим выборку на тренировочную и валидационную

In [13]:

```
train_df = df.sample(frac=0.8, random_state=42)
val_df = df.drop(train_df.index)
```

In [14]:

```
MAX_LEN = 1150
N_GROUPS = 203
```

In [15]:

```
import numpy as np
import math
from keras.utils import Sequence

def group_to_vec_w2v(n):
    try:
        vec = wv_model.wv[str(n)]
    except:
        vec = np.zeros(50)
    return vec

class DataLoader(Sequence):
    def __init__(self, df, batch_size, group_to_vec=group_to_vec_w2v, is_test=False
        self.df = df
        self.batch_size = batch_size
        self.group_to_vec = group_to_vec
        self.is_test = is_test

    def __len__(self):
        return math.ceil(len(self.df) / self.batch_size)

    def _preproc_data(self, df):
        result = []
        for _, row in df.iterrows():
            row_data = [(*self.group_to_vec(group), amount) for date, group, amount
            for _ in range(MAX_LEN-len(row_data)):
                row_data.append([0] * (50+1))
            result.append(row_data)
        result = np.array(result)
        return result

    def on_epoch_end(self):
        self.df = self.df.sample(frac=1)

    def __getitem__(self, idx):
        batch_df = self.df[idx * self.batch_size: (idx + 1) * self.batch_size]
        batch_X = self._preproc_data(batch_df)
        if not self.is_test:
            batch_y_values = batch_df['bins'].values
            batch_y = np.zeros((len(batch_y_values), 4), dtype=np.int8)
            for i, v in enumerate(batch_y_values):
                batch_y[i][v] = 1
            return batch_X, batch_y
        else:
            return batch_X
```

```
Using TensorFlow backend.
```

In [16]:

```
w2v_train_loader = DataLoader(train_df, batch_size=32, group_to_vec=group_to_vec_w2
w2v_val_loader = DataLoader(val_df, batch_size=32, group_to_vec=group_to_vec_w2v)
```

In [17]:

```
from keras.models import Model, Sequential
from keras.layers import Input, BatchNormalization, Dropout, Dense, Bidirectional

import keras as K

from keras.optimizers import Adam, RMSprop, SGD
from keras.layers import LSTM

def get_model(input_shape):
    data = Input(shape=input_shape, name='data')

    lstm_1 = Bidirectional(LSTM(50, return_sequences=False))

    x = lstm_1(data)
    x = BatchNormalization()(x)
    x = Dense(10, activation='relu')(x)
    x = BatchNormalization()(x)
    output = Dense(4)(x)

    model = Model(
        inputs=[data],
        outputs=[output]
    )
    model.compile(
        optimizer=Adam(learning_rate=0.001),
        loss=K.losses.CategoricalCrossentropy(
            from_logits=True
        ),
        metrics=['accuracy']
    )
    return model
```

In [18]:

```
w2v_emb_model = get_model(input_shape=(1150, 51))
print(w2v_emb_model.summary())

reduce_lr = K.callbacks.ReduceLROnPlateau(monitor='val_loss',
                                          factor=0.5,
                                          patience=3,
                                          min_lr=0.0001,
                                          verbose=1)
early_stop = K.callbacks.EarlyStopping(monitor='val_loss',
                                       patience=10,
                                       verbose=1,
                                       restore_best_weights=True)
w2v_emb_model.fit(w2v_train_loader,
          validation_data=w2v_val_loader,
          epochs=100,
          workers=10,
          callbacks=[reduce_lr, early_stop],
          verbose=1)
```

```
Epoch 27/100
600/600 [==============================] - 516s 860ms/step - loss:
0.8148 - accuracy: 0.6378 - val_loss: 0.9420 - val_accuracy: 0.5985
Epoch 28/100
  1/600 [..............................] - ETA: 8:33 - loss: 0.8097
- accuracy: 0.6250

/home/kirlev/Projects/Python/venv/lib/python3.6/site-packages/kera
s/callbacks/callbacks.py:95: RuntimeWarning: Method (on_train_batch
_end) is slow compared to the batch update (1.212100). Check your c
allbacks.
  % (hook_name, delta_t_median), RuntimeWarning)

600/600 [==============================] - 513s 855ms/step - loss:
0.8163 - accuracy: 0.6414 - val_loss: 0.8187 - val_accuracy: 0.6000
Restoring model weights from the end of the best epoch
Epoch 00028: early stopping
```

Out[18]:

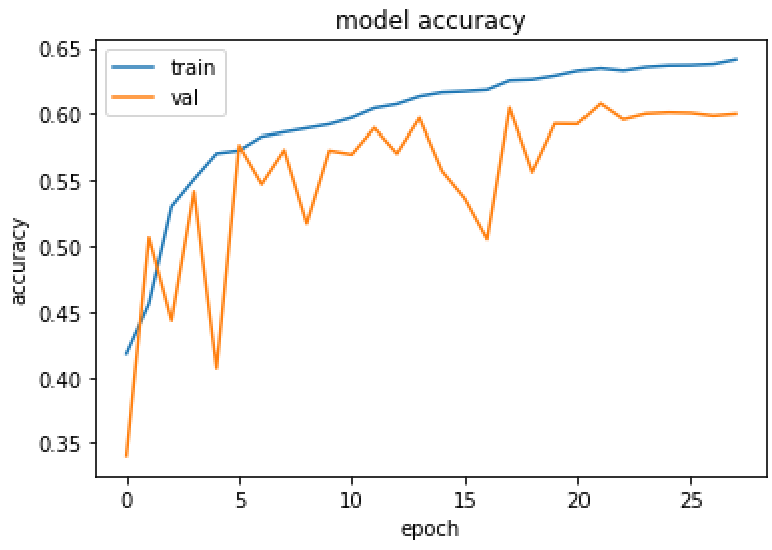
```
<keras.callbacks.callbacks.History at 0x7f80d8e1ac88>
```

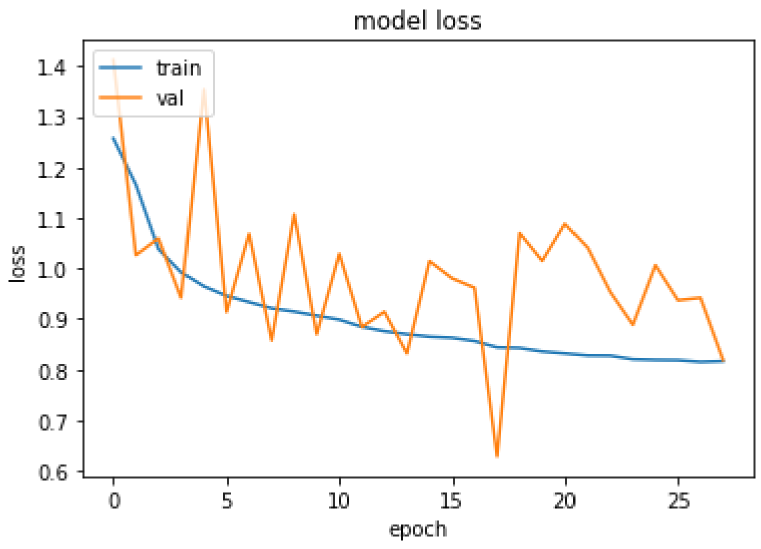
In [19]:

```
w2v_emb_model.save('model_w2v.h5')
```

In [28]:

```
from matplotlib import pyplot as plt
# summarize history for accuracy
plt.plot(w2v_emb_model.history.history['accuracy'])
plt.plot(w2v_emb_model.history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(w2v_emb_model.history.history['loss'])
plt.plot(w2v_emb_model.history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```





Let's test!

In [20]:

```
test_df = trans_agg.merge(test_target)
```

In [21]:

```
w2v_test_loader = DataLoader(test_df, batch_size=32, group_to_vec=group_to_vec_w2v,
```

In [22]:

```
preds = w2v_emb_model.predict(w2v_test_loader)
```

In [23]:

```
from sklearn.metrics import accuracy_score
accuracy_score(preds.argmax(axis=1), test_df['bins'])  # should be ~0.6
```

Out[23]:

```
0.5938333333333333
```