Creating temporal features:

The following code is to create a temporal dataset, for analyzing the temporal behaviour of the customers. 3 features were picked: value, frequency and quantity purchased. The value and quantity feature was aggregated using the SUM function, and the frequency was derived by aggregating the distinct receipt\_id.

sql\_top = """

SELECT customer\_id,

%(ref\_date)s::DATE AS ref\_day,

{0}(DISTINCT CASE WHEN purchased\_at::DATE > %(ref\_date)s::DATE AND purchased\_at::DATE <= %(ref\_date)s::DATE + %(ows)s::INT THEN ABS(receipt\_id) ELSE 0 END) <> 0 THEN 1 ELSE 0 END as output\_feature,

""".format(output\_fun)

sql = sql\_top

for i in range(0,num\_periods):

sql += "{2}(CASE WHEN purchased\_at::DATE > %(ref\_date)s::DATE -%(ws)s::INT\*({0}+1) AND purchased\_at::DATE <= %(ref\_date)s::DATE-%(ws)s::INT\*({0}) THEN {3} ELSE 0 END) as vf{1},\n".format(i, i+1, window\_agg\_fun, "value")

for i in range(0,num\_periods):

sql += "{2} (DISTINCT CASE WHEN purchased\_at::DATE > %(ref\_date)s::DATE -%(ws)s::INT\*({0}+1) AND purchased\_at::DATE <= %(ref\_date)s::DATE-%(ws)s::INT\*({0}) THEN qty ELSE 0 END) as qf{1},\n".format(i, i+1, window\_agg\_fun2)

for i in range(0,num\_periods):

sql += "{2} (Distinct CASE WHEN purchased\_at::DATE > %(ref\_date)s::DATE -%(ws)s::INT\*({0}+1) AND purchased\_at::DATE <= %(ref\_date)s::DATE-%(ws)s::INT\*({0}) THEN ABS(receipt\_id) END) as freq\_f{1},\n".format(i, i+1, window\_agg\_fun3)

sql\_bottom = """

FROM ml16.receipts join ml16.receipt\_lines using(receipt\_id)

where purchased\_at > ('2018-06-13 00:00:00')

GROUP BY customer\_id

having count(purchased\_at) > 1

"""

sql = sql[:-2] + sql\_bottom

with psycopg2.connect("host='{}' dbname='nlab' user='{}' password='{}'".format(db\_ip, user, pw)) as conn:

df = pd.read\_sql(sql, conn, params = {'ref\_date':reference\_day, 'ws':tumbling\_window\_size, 'ows':output\_window\_size})

df.set\_index('customer\_id')

input\_feature = df.drop(columns = ['ref\_day','output\_feature'], inplace = False)

return input\_feature, df.output\_feature

Final predictor:

The final prediction model is as below, in order to calculate the churn rate of only the active customer, a condition is put in test and training set, where only those customers are considered whose frequency of visit is greater than 1:

from sklearn import metrics

from datetime import date, timedelta

import datetime

from sklearn.ensemble import RandomForestClassifier

from imblearn.over\_sampling import SMOTE

tumbling\_window\_size = 7

output\_window\_size = 27

total\_holdout = 8

now= '2019-11-19'

now\_day= datetime.datetime.strptime(now, '%Y-%m-%d').date()

model = RandomForestClassifier(n\_estimators = 100)

for i in range(total\_holdout):

test\_x, test\_y = get\_dataset(str(now\_day - timedelta(days=output\_window\_size)), tumbling\_window\_size, output\_window\_size)

test\_x= test\_x.loc[test\_x['freq\_f1'] > 0]

ind = list(test\_x.index.values)

test\_y = test\_y.loc[test\_y.index.isin(ind)]

train\_x, train\_y = get\_dataset(str(now\_day - 2\*timedelta(days=output\_window\_size)), tumbling\_window\_size, output\_window\_size)

train\_x= train\_x.loc[train\_x['freq\_f1'] > 0]

ind = list(train\_x.index.values)

train\_y=train\_y.loc[train\_y.index.isin(ind)]

model.fit(train\_x, train\_y)

preds = model.predict(test\_x)

print(model.predict(test\_x))

Feature Importance:

The permutation function is used for deriving the importance of the features from the temporal dataset.

import eli5

from eli5.sklearn import PermutationImportance

feature\_importance\_scores = {}

X\_train\_sub, X\_test\_sub, y\_train\_sub, y\_test\_sub = train\_test\_split(X\_train, y\_train, test\_size=0.33, random\_state=42)

rf\_perm = RandomForestClassifier()

rf\_perm.fit(X\_train\_sub, y\_train\_sub)

perm = PermutationImportance(rf\_perm, cv='prefit')

perm.fit(X\_test\_sub, y\_test\_sub)

y\_pred = rf\_perm.predict(X\_test)

scores = accuracy\_score(y\_test, y\_pred)

method\_name = 'Perm non-cv RF ({:.2f})'.format(scores\*100)

feature\_importance\_scores[method\_name] = perm.feature\_importances\_

print\_variable\_importances( X\_train.columns, feature\_importance\_scores )