In [1]:
```
!pip3 install pandas --quiet
!pip3 install statsmodels --quiet
!pip3 install numpy --quiet
!pip3 install matplotlib --quiet
!pip3 install seaborn --quiet
!pip3 install sklearn --quiet
```

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.linear_model import SGDClassifier, LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import cross_val_score, GridSearchCV
from collections import defaultdict
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import OrdinalEncoder, LabelEncoder, MinMaxScaler, StandardScaler
from sklearn.svm import SVC
from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import roc_auc_score
```

In [3]:
```
ain_1 = pd.read_csv('train_1.csv', delimiter=';', dtype={'SUM_TRANS':'float'}, decimal=',', parse_dates=
ain_2 = pd.read_csv('train_2.csv', delimiter=';', dtype={'INCOME_MAIN_AMT':'float'}, decimal=',', low_mem
```

In [4]:
```
;t_1 = pd.read_csv('test_1.csv', delimiter=';', dtype={'SUM_TRANS':'float'}, decimal=',', parse_dates=[2]
;t_2 = pd.read_csv('test_2.csv', delimiter=';', dtype={'INCOME_MAIN_AMT':'float'}, decimal=',', low_memor
```

In [5]:
```
train = train_1.drop(['PROD_TYPE', 'SUM_TRANS', 'LOCATION_NAME'], axis=1)
```

In [6]:
```
train.set_index('TRANS_DTTM', inplace=True)
```

Выделение признаков

In [7]:
```
sers = train['ID'].unique()

xtracted_features = pd.DataFrame(columns=['shift_1', 'shift_2', 'shift_3', 'rolling_3', 'dayofweek', 'mo

or user in users[:10]:
    user_mcc_code = train.loc[train['ID'] == user, 'MCC_CODE'].unique()
    for mcc_code in user_mcc_code:
        temp_df = train.loc[((train['ID'] == user) & (train['MCC_CODE'] == mcc_code)), 'ID']
        temp_df.sort_index(inplace=True)

        resampled_df = pd.DataFrame(temp_df.resample('1D').count())

        resampled_df['shift_1'] = resampled_df.shift(1, fill_value=0)
        resampled_df['shift_2'] = resampled_df['ID'].shift(2, fill_value=0)
        resampled_df['shift_3'] = resampled_df['ID'].shift(3, fill_value=0)
        resampled_df['rolling_3'] = resampled_df['ID'].shift().rolling(3).mean()
        resampled_df['dayofweek'] = resampled_df.index.dayofweek
        resampled_df['month'] = resampled_df.index.month
        resampled_df['mcc_code'] = np.array([mcc_code] * resampled_df.shape[0])
        resampled_df['id'] = np.array([user] * resampled_df.shape[0])

        extracted_features = pd.concat([extracted_features, resampled_df], ignore_index=True)
```

In [8]:
```
extracted_features = extracted_features.fillna(0)

scaler = StandardScaler()
scaled_data = scaler.fit_transform(extracted_features.drop(['id', 'mcc_code'], axis=1))
```

In [9]:
```
scaled_extracted = pd.DataFrame(columns=['shift_1', 'shift_2', 'shift_3', 'rolling_3', 'dayofweek', 'mon
scaled_extracted['user_id'] = extracted_features['id']
```

In [10]:
```
lbl_enc = LabelEncoder()
scaled_extracted['labl_user_id'] = lbl_enc.fit_transform(scaled_extracted['user_id'])
```

In [11]:
```
features = scaled_extracted.drop('user_id', axis=1)
targets = extracted_features['mcc_code']
```

Обучение классификатора

In [12]:
```
%%time
tsv = TimeSeriesSplit()
grb = GradientBoostingClassifier(n_estimators=500, max_depth=3, learning_rate=.8)
scores = cross_val_score(grb, features, targets, cv=3, n_jobs=-1)
```

```
/opt/conda/lib/python3.9/site-packages/sklearn/model_selection/_split.py:676: UserWarning: The least po
pulated class in y has only 1 members, which is less than n_splits=3.
  warnings.warn(

CPU times: user 22.8 ms, sys: 136 ms, total: 159 ms
Wall time: 4min 31s
```

In [13]:
```
grb = GradientBoostingClassifier(n_estimators=600, max_depth=3, learning_rate=.8)
grb.fit(features, targets)
```

Out[13]:
```
GradientBoostingClassifier(learning_rate=0.8, n_estimators=600)
```

In [22]:
```
grb
```

Out[22]:
```
GradientBoostingClassifier(learning_rate=0.8, n_estimators=600)
```

Подбор релевантной корзины клиенту

In [14]:
```
test = test_1.drop(['PROD_TYPE', 'SUM_TRANS', 'LOCATION_NAME'], axis=1)
test.set_index('TRANS_DTTM', inplace=True)
```

In [15]:
```
ısers = train['ID'].unique()

ɛxtracted_features = pd.DataFrame(columns=['shift_1', 'shift_2', 'shift_3', 'rolling_3', 'dayofweek', 'mo

ıser = 500000002152261401

ıser_mcc_code = test.loc[test['ID'] == user, 'MCC_CODE'].unique()
for mcc_code in user_mcc_code:
    temp_df = test.loc[((test['ID'] == user) & (test['MCC_CODE'] == mcc_code)), 'ID']
    temp_df.sort_index(inplace=True)

    resampled_df = pd.DataFrame(temp_df.resample('1D').count())

    resampled_df['shift_1'] = resampled_df.shift(1, fill_value=0)
    resampled_df['shift_2'] = resampled_df['ID'].shift(2, fill_value=0)
    resampled_df['shift_3'] = resampled_df['ID'].shift(3, fill_value=0)
    resampled_df['rolling_3'] = resampled_df['ID'].shift().rolling(3).mean()
    resampled_df['dayofweek'] = resampled_df.index.dayofweek
    resampled_df['month'] = resampled_df.index.month
    resampled_df['mcc_code'] = np.array([mcc_code] * resampled_df.shape[0])
    resampled_df['id'] = np.array([user] * resampled_df.shape[0])

    extracted_features = pd.concat([extracted_features, resampled_df], ignore_index=True)
```

In [16]:
```
extracted_features = extracted_features.fillna(0)

scaled_data = scaler.fit_transform(extracted_features.drop(['id', 'mcc_code'], axis=1))
```

In [17]:
```
led_extracted = pd.DataFrame(columns=['shift_1', 'shift_2', 'shift_3', 'rolling_3', 'dayofweek', 'month'
led_extracted['user_id'] = extracted_features['id']
```

In [26]:
```
scaled_extracted['labl_user_id'] = lbl_enc.fit_transform(scaled_extracted['user_id'])
```

In [27]:
```
features = scaled_extracted.drop('user_id', axis=1)
targets = extracted_features['mcc_code']
```

In [28]:
```
user_predicts = grb.predict(features)
```

Определение расстояния до ближайшего офлайн mcc

In [29]:
```
def gen_coordinate(train):
    return np.random.randint(0, 1023, size=(train.shape[0], 1))
```

In [30]:
```python
#условные координаты оффлайн mcc
train_1['coordinates_x'] = gen_coordinate(train_1)
train_1['coordinates_y'] = gen_coordinate(train_1)
train_1['distance'] = 0

user_vector = [123, 100] #условный вектор координат юзера
train_1['euclidean_distance'] = np.sqrt(
    (train_1['coordinates_x'] - user_vector[0])**2
    + (train_1['coordinates_y'] - user_vector[1])**2
)
```

In [31]:
```python
#условные координаты оффлайн mcc
test_1['coordinates_x'] = gen_coordinate(test_1)
test_1['coordinates_y'] = gen_coordinate(test_1)
test_1['distance'] = 0

user_vector = [123, 100] #условный вектор координат юзера
test_1['euclidean_distance'] = np.sqrt(
    (test_1['coordinates_x'] - user_vector[0])**2
    + (test_1['coordinates_y'] - user_vector[1])**2
)
```

In [34]:
```python
def get_nearest_mcc(data, user_id, recommendations, less_distance=1024):
    return data.loc[
        (data['ID'] == user_id)
        & (data['euclidean_distance'] < less_distance)
        & (data['MCC_CODE'].isin(recommendations))
    ]
```

In [35]:
```python
#отфильтрованные рекомендации, выдаваемые клиенту
get_nearest_mcc(test_1, 500000002152261401, user_predicts)
```

Out[35]:

ID	PROD_TYPE	TRANS_DTTM	MCC_CODE	SUM_TRANS	LOCATION_NAME	coordinates_x	coordinates_y	distance	eu
02152261401	2	2021-09-29 13:59:03	5411	651.82	MONETKA\11 YUNOSTI STR\MEZHDURECHENS\652877 ...	748	99	0	
02152261401	2	2021-08-21 06:38:06	5411	714.64	NaN	501	106	0	
02152261401	2	2021-09-27 17:04:57	5411	479.86	NaN	680	780	0	
02152261401	2	2021-09-28 16:16:11	5411	698.92	NaN	908	565	0	
02152261401	2	2021-08-22 17:14:36	5411	471.84	NaN	9	388	0	

Определение категории клиентов

In [36]:
```python
def gen_category(row):
    if row['INCOME_MAIN_AMT'] < 30000:
        return '<30000'
    elif row['INCOME_MAIN_AMT'] < 50000:
        return '<50000'
    elif row['INCOME_MAIN_AMT'] < 80000:
        return '<80000'
    elif row['INCOME_MAIN_AMT'] < 120000:
        return '<120000'
    elif row['INCOME_MAIN_AMT'] < 180000:
        return '<180000'
    elif row['INCOME_MAIN_AMT'] < 270000:
        return '<270000'
    elif row['INCOME_MAIN_AMT'] < 390000:
        return '<390000'
    elif row['INCOME_MAIN_AMT'] < 630000:
        return '<630000'
    else:
        return '>=630000'
```

In [37]:
```python
train_2['category'] = train_2.apply(gen_category, axis=1)
test_2['category'] = test_2.apply(gen_category, axis=1)
```

Определение сходства клиентов

In [38]:
```python
egated_by_user_mcc = train_1[train_1['ID'].isin(users[:10])].pivot_table(index='ID', columns='MCC_CODE',
_matrix = aggregated_by_user_mcc.T.corr()
```

На основе матрицы сходства можно искать максимлально похожих юзеров (скажем, где коэфициент корреляции Пирсона >.9). Получать множества товаров всех похожих юзеров и искать между ними разность множеств. Разность множеств дополнительно фильтруем по категориям платёжеспособности активного клиента и выдаём, как предложение по расширению корзины

In [40]:
```
corr_matrix > .99
```

Out[40]:

ID	500000000004725733	500000000050139448	500000000158893444	500000000402535207	500000000608267511	50000000063
ID						
500000000004725733	True	False	False	False	False	
500000000050139448	False	True	False	False	False	
500000000158893444	False	False	True	False	False	
500000000402535207	False	False	False	True	False	
500000000608267511	False	False	False	False	True	
500000000634517647	False	False	False	True	False	
500000001089710588	False	False	False	False	False	
500000001271933224	False	False	False	False	False	
500000001639102687	False	False	False	False	False	
500000003407797504	False	False	False	False	False	

In [51]:
```
corr_matrix[500000000634517647][corr_matrix[500000000634517647] > .98]
```

Out[51]:
```
ID
500000000402535207    0.991731
500000000608267511    0.981278
500000000634517647    1.000000
Name: 500000000634517647, dtype: float64
```

In [ ]: