In [1]:
```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
fts_svm = pd.read_excel('./data/features.xlsx', header=None)
fts_svm = np.array(fts_svm)
fts_svm = [x[0].split()[1] for x in fts_svm]
print(len(fts_svm))
print(fts_svm)
```

```
26
['dep_bl_3am_svm', 'inv_bl_3am_svm', 'cr_bl_3am_svm', 'oprd_bl_3am_svm', 'dep
_oacc_ct_svm', 'ira_oacc_ct_svm', 'inv_oacc_ct_svm', 'meac_oacc_ct_svm', 'mes
d_oacc_ct_svm', 'fsvc_oacc_ct_svm', 'cred_oacc_ct_svm', 'tma_chnl_dc_ct_svm',
'tma_chnl_cc_ct_svm', 'tma_chnl_bcplt_ct_svm', 'tma_chnl_bctlr_ct_svm', 'tma_
chnl_atm_ct_svm', 'tma_chnl_olb_ct_svm', 'tma_chnl_mob_ct_svm', 'tma_chnl_ach
_ct_svm', 'tma_chnl_icc_ct_svm', 'tma_chnl_dcc_ct_svm', 'tma_chnl_mcc_ct_sv
m', 'tma_chnl_ccc_ct_svm', 'chnl_seg2_svm', 'prd_cat_svm', 'ttl_cmp_svm']
```

In [3]:
```
fts_omp = [x.replace('svm', 'omp') for x in fts_svm]
fts_tmp = [x.replace('svm', 'tmp') for x in fts_svm]
fts_smp = [x.replace('svm', 'smp') for x in fts_svm]
```

In [4]:
```
fts = list()
fts = fts_svm + fts_omp + fts_tmp + fts_smp
fts.append('vint_dt')
fts.append('rwd_tier_dt')
fts.append('pr_enrll_any')
print('Total number of features: %d' % (len(fts)))
```

```
Total number of features: 107
```

In [5]:
```
df = pd.read_csv('./data/customer_data.csv', header=0, index_col=0)
```

In [7]:
```
df_filtered = df[fts]
df_filtered.shape
```

Out[7]:
```
(127239, 107)
```

In [8]:
```
from sklearn.utils import shuffle

df_filtered = shuffle(df_filtered,random_state=42).reset_index(drop=True)
df_filtered.head()
```

Out[8]:

	dep_bl_3am_svm	inv_bl_3am_svm	cr_bl_3am_svm	oprd_bl_3am_svm	dep_oacc_ct_svm	ira_
0	63835.12	0.0	728.21	0	2	
1	19838.90	0.0	0.00	0	2	
2	16313.79	0.0	58.61	0	3	
3	10385.28	0.0	0.00	0	1	
4	14493.99	0.0	6880.69	0	2	

5 rows × 107 columns

In [9]:
```
ftype = dict()
ftype['cat'] = ['chnl_seg2_omp', 'prd_cat_omp', 'chnl_seg2_tmp', 'prd_cat_tmp'
, 'chnl_seg2_smp', 'prd_cat_smp']
ftype['quant'] = ['dep_bl_3am_omp', 'inv_bl_3am_omp', 'cr_bl_3am_omp', 'tma_ch
nl_dc_ct_omp', 'tma_chnl_cc_ct_omp', \
                  'tma_chnl_bcplt_ct_omp', 'tma_chnl_bctlr_ct_omp', 'tma_chnl_
atm_ct_omp', 'tma_chnl_olb_ct_omp', \
                  'tma_chnl_mob_ct_omp', 'tma_chnl_ach_ct_omp', 'dep_bl_3am_tm
p', 'inv_bl_3am_tmp', \
                  'cr_bl_3am_tmp', 'tma_chnl_dc_ct_tmp', 'tma_chnl_cc_ct_tmp',
'tma_chnl_bcplt_ct_tmp', \
                  'tma_chnl_bctlr_ct_tmp', 'tma_chnl_atm_ct_tmp', 'tma_chnl_ol
b_ct_tmp', 'tma_chnl_mob_ct_tmp', \
                  'tma_chnl_ach_ct_tmp', 'dep_bl_3am_smp', 'inv_bl_3am_smp',
'cr_bl_3am_smp', 'tma_chnl_dc_ct_smp', \
                  'tma_chnl_cc_ct_smp', 'tma_chnl_bcplt_ct_smp', 'tma_chnl_bct
lr_ct_smp', 'tma_chnl_atm_ct_smp', \
                  'tma_chnl_olb_ct_smp', 'tma_chnl_mob_ct_smp', 'tma_chnl_ach_
ct_smp']
```

In [10]:
```
# Label Encode Categorical Featrues
from sklearn import preprocessing

def clean(input_df):
    df = input_df.copy()

    for col in ftype['cat']:
        le = preprocessing.LabelEncoder()
        df[str(col)+'_Encoded'] = le.fit_transform(df[col].astype(str))

        del df[col]

    return df
```

In [11]:
```
from sklearn.preprocessing import MinMaxScaler
from scipy.stats import skew

def preprocess(input_df,scale,onehot):
    df = input_df.copy()

    if scale:
        # scaling the quantitative features
        scaler = MinMaxScaler()
        df[ftype['quant']] = scaler.fit_transform(df[ftype['quant']])

    # Convert to one-hot Encoding
    if onehot:
        encoded_features = [x + '_Encoded' for x in ftype['cat']]

        onehotted = pd.get_dummies(data=df, columns=encoded_features)

        return onehotted
    else:
        return df
```

In [12]:
```
model_data = df_filtered.copy().drop(['vint_dt', 'rwd_tier_dt'], axis=1)
```

In [13]:
```
model_data_clean = clean(model_data)
model_data_preprocessed = preprocess(model_data_clean, scale=True, onehot=True
).dropna()
```

In [14]:
```
y = model_data_preprocessed['pr_enrll_any']
X = model_data_preprocessed.drop(['pr_enrll_any', ], axis=1)
```

In [15]:
```
le = preprocessing.LabelEncoder()
y_encoded = le.fit_transform(y.astype(str))
```

In [16]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.
2)
```

In [17]:
```
def get_features(id_str):
    cols = model_data_preprocessed.columns

    new_cols = list()
    for col in cols:
        splits = col.split('_')
        if id_str in splits:
            new_cols.append(col)

    return new_cols
```

In [18]:
```
new_fts_omp = get_features('omp')
new_fts_tmp = get_features('tmp')
new_fts_smp = get_features('smp')
```

In [19]:
```
X_train_omp, X_test_omp = X_train[new_fts_omp], X_test[new_fts_omp]
X_train_tmp, X_test_tmp = X_train[new_fts_tmp], X_test[new_fts_tmp]
X_train_smp, X_test_smp = X_train[new_fts_smp], X_test[new_fts_smp]
```

In [20]:
```
y.value_counts()
```

Out[20]:
```
N    73880
Y    53359
Name: pr_enrll_any, dtype: int64
```

Logistic Regression

OMP

In [21]:
```
from sklearn.linear_model import LogisticRegressionCV
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_curve
from sklearn.metrics import confusion_matrix
```

In [22]:
```
lr_omp = LogisticRegressionCV(cv=5, random_state=0, Cs=[1, 1e2, 1e3, 1e4, 1e5,
1e6, 1e7], verbose=2).fit(X_train_omp, y_train)
print(lr_omp.C_)
print(lr_omp)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worke
rs.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    6.4s remaining:    0.
0s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:   30.9s finished

[10000000.]
LogisticRegressionCV(Cs=[1, 100.0, 1000.0, 10000.0, 100000.0, 1000000.0, 1000
0000.0],
           class_weight=None, cv=5, dual=False, fit_intercept=True,
           intercept_scaling=1.0, max_iter=100, multi_class='warn',
           n_jobs=None, penalty='l2', random_state=0, refit=True,
           scoring=None, solver='lbfgs', tol=0.0001, verbose=2)
```

In [23]:
```
preds_lr_omp = lr_omp.predict(X_test_omp)
print(classification_report(y_test, preds_lr_omp))
```

```
              precision    recall  f1-score   support

           0       0.68      0.82      0.74     14728
           1       0.65      0.47      0.55     10720

   micro avg       0.67      0.67      0.67     25448
   macro avg       0.66      0.64      0.64     25448
weighted avg       0.67      0.67      0.66     25448
```

In [24]:
```
print(accuracy_score(y_test, preds_lr_omp))
```

```
0.6699544168500472
```

In [25]:
```
print(confusion_matrix(y_test, preds_lr_omp))
```

```
[[12006  2722]
 [ 5677  5043]]
```

TMP

In [26]:
```
lr_tmp = LogisticRegressionCV(cv=5, random_state=0, Cs=[1e-4, 1e-3, 1e-2, 1e-1
, 1, 1e2, 1e3, 1e4]).fit(X_train_tmp, y_train)
print(lr_tmp.C_)
print(lr_tmp)
```

```
[10000.]
LogisticRegressionCV(Cs=[0.0001, 0.001, 0.01, 0.1, 1, 100.0, 1000.0, 10000.
0],
           class_weight=None, cv=5, dual=False, fit_intercept=True,
           intercept_scaling=1.0, max_iter=100, multi_class='warn',
           n_jobs=None, penalty='l2', random_state=0, refit=True,
           scoring=None, solver='lbfgs', tol=0.0001, verbose=0)
```

In [27]:
```
preds_lr_tmp = lr_tmp.predict(X_test_tmp)
print(classification_report(y_test, preds_lr_tmp))
```

```
              precision    recall  f1-score   support

           0       0.66      0.81      0.73     14728
           1       0.63      0.43      0.51     10720

   micro avg       0.65      0.65      0.65     25448
   macro avg       0.65      0.62      0.62     25448
weighted avg       0.65      0.65      0.64     25448
```

In [28]:
```
print(accuracy_score(y_test, preds_lr_tmp))
```

```
0.6534894687205282
```

SMP

In [29]:
```
lr_smp = LogisticRegressionCV(cv=5, random_state=0, Cs=[1e-4, 1e-3, 1e-2, 1e-1
, 1, 1e2, 1e3, 1e4]).fit(X_train_smp, y_train)
print(lr_smp.C_)
print(lr_smp)
```

```
[10000.]
LogisticRegressionCV(Cs=[0.0001, 0.001, 0.01, 0.1, 1, 100.0, 1000.0, 10000.
0],
           class_weight=None, cv=5, dual=False, fit_intercept=True,
           intercept_scaling=1.0, max_iter=100, multi_class='warn',
           n_jobs=None, penalty='l2', random_state=0, refit=True,
           scoring=None, solver='lbfgs', tol=0.0001, verbose=0)
```

In [30]:
```
preds_lr_smp = lr_smp.predict(X_test_smp)
print(classification_report(y_test, preds_lr_smp))
```

```
              precision    recall  f1-score   support

           0       0.66      0.80      0.72     14728
           1       0.61      0.43      0.51     10720

   micro avg       0.64      0.64      0.64     25448
   macro avg       0.63      0.62      0.61     25448
weighted avg       0.64      0.64      0.63     25448
```

In [31]:
```
print(accuracy_score(y_test, preds_lr_smp))
```

```
0.6443728387299591
```

In [ ]:
```

```

Quadratic Discriminant Analysis

In [32]:
```
import time
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
```

OMP

In [33]:
```
start_time = time.time()
qda_omp = QuadraticDiscriminantAnalysis(reg_param=1e-10).fit(X_train_omp, y_tr
ain)
elapsed_time = time.time() - start_time

print(elapsed_time)
print(qda_omp)
```

```
0.3268768787384033
QuadraticDiscriminantAnalysis(priors=None, reg_param=1e-10,
               store_covariance=False, store_covariances=None, tol=0.0001)
```

In [34]:
```
preds_qda_omp = qda_omp.predict(X_test_omp)
print(classification_report(y_test, preds_qda_omp))
```

```
              precision    recall  f1-score   support

           0       0.67      0.71      0.69     14728
           1       0.56      0.51      0.54     10720

   micro avg       0.63      0.63      0.63     25448
   macro avg       0.62      0.61      0.61     25448
weighted avg       0.62      0.63      0.63     25448
```

In [35]:
```
print(accuracy_score(y_test, preds_qda_omp))
```

```
0.6279864822382899
```

In [38]:
```
preds_qda_omp = qda_omp.predict(X_test_omp)
print(classification_report(y_test, preds_qda_omp))
print(accuracy_score(y_test, preds_qda_omp))
```

```
              precision    recall  f1-score   support

           0       0.67      0.71      0.69     14728
           1       0.56      0.51      0.54     10720

   micro avg       0.63      0.63      0.63     25448
   macro avg       0.62      0.61      0.61     25448
weighted avg       0.62      0.63      0.63     25448

0.6279864822382899
```

In [39]:
```
print(confusion_matrix(y_test, preds_qda_omp))
```

```
[[10478  4250]
 [ 5217  5503]]
```

TMP

In [40]:
```
start_time = time.time()
qda_tmp = QuadraticDiscriminantAnalysis(reg_param=0.1).fit(X_train_tmp, y_trai
n)
elapsed_time = time.time() - start_time

print(elapsed_time)
print(qda_tmp)
```

```
0.2063617706298828
QuadraticDiscriminantAnalysis(priors=None, reg_param=0.1,
               store_covariance=False, store_covariances=None, tol=0.0001)
```

In [41]:
```
preds_qda_tmp = qda_tmp.predict(X_test_tmp)
print(classification_report(y_test, preds_qda_tmp))
```

```
              precision    recall  f1-score   support

           0       0.62      0.83      0.71     14728
           1       0.57      0.31      0.40     10720

   micro avg       0.61      0.61      0.61     25448
   macro avg       0.60      0.57      0.56     25448
weighted avg       0.60      0.61      0.58     25448
```

In [42]:
```
print(accuracy_score(y_test, preds_qda_tmp))
```

```
0.6119537881169443
```

SMP

In [43]:
```
start_time = time.time()
qda_smp = QuadraticDiscriminantAnalysis(reg_param=0.1).fit(X_train_smp, y_trai
n)
elapsed_time = time.time() - start_time

print(elapsed_time)
print(qda_smp)
```

```
0.21283984184265137
QuadraticDiscriminantAnalysis(priors=None, reg_param=0.1,
               store_covariance=False, store_covariances=None, tol=0.0001)
```

In [44]:
```
preds_qda_smp = qda_smp.predict(X_test_smp)
print(classification_report(y_test, preds_qda_smp))
```

```
              precision    recall  f1-score   support

           0       0.62      0.85      0.71     14728
           1       0.57      0.27      0.37     10720

   micro avg       0.61      0.61      0.61     25448
   macro avg       0.59      0.56      0.54     25448
weighted avg       0.60      0.61      0.57     25448
```

In [45]:
```
print(accuracy_score(y_test, preds_qda_smp))
```

```
0.6061380069160641
```

In [ ]:
```

```

In [47]:
```
# d = {'y': y_test, 'lr_omp': probs_lr_omp, 'lr_tmp': probs_lr_tmp, 'lr_smp':
 probs_lr_smp, \
#     'qda_omp': probs_qda_omp, 'qda_tmp': probs_qda_tmp, 'qda_smp': probs_qda
_smp}
# results = pd.DataFrame(data=d)
```

In [48]:
```
# results.to_csv('./data/model_probs.csv', sep=',', header=True)
```

Neural Network

In [49]:
```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
```

OMP

In [50]:
```
nn_omp = MLPClassifier(activation='relu', hidden_layer_sizes=(100,80), verbose
=2, max_iter=300).fit(X_train_omp, y_train)
```

```
Iteration 1, loss = 0.62574851
Iteration 2, loss = 0.61455946
Iteration 3, loss = 0.60999661
Iteration 4, loss = 0.60684630
Iteration 5, loss = 0.60383979
Iteration 6, loss = 0.59941954
Iteration 7, loss = 0.59496380
Iteration 8, loss = 0.59125224
Iteration 9, loss = 0.58302421
Iteration 10, loss = 0.57717235
Iteration 11, loss = 0.56847180
Iteration 12, loss = 0.56223276
Iteration 13, loss = 0.55708374
Iteration 14, loss = 0.55103018
Iteration 15, loss = 0.54661040
Iteration 16, loss = 0.54104959
Iteration 17, loss = 0.53928548
Iteration 18, loss = 0.53686639
Iteration 19, loss = 0.53090583
Iteration 20, loss = 0.52794420
Iteration 21, loss = 0.52747173
Iteration 22, loss = 0.52367177
Iteration 23, loss = 0.52144593
Iteration 24, loss = 0.52116406
Iteration 25, loss = 0.52021359
Iteration 26, loss = 0.51912364
Iteration 27, loss = 0.51892369
Iteration 28, loss = 0.51720830
Iteration 29, loss = 0.51544101
Iteration 30, loss = 0.51543417
Iteration 31, loss = 0.51477391
Iteration 32, loss = 0.51297365
Iteration 33, loss = 0.51580459
Iteration 34, loss = 0.51430360
Iteration 35, loss = 0.51343292
Iteration 36, loss = 0.51664702
Iteration 37, loss = 0.51136144
Iteration 38, loss = 0.51083723
Iteration 39, loss = 0.51031666
Iteration 40, loss = 0.51135870
Iteration 41, loss = 0.51179240
Iteration 42, loss = 0.50958389
Iteration 43, loss = 0.50819220
Iteration 44, loss = 0.50964662
Iteration 45, loss = 0.50943377
Iteration 46, loss = 0.50954898
Iteration 47, loss = 0.50762634
Iteration 48, loss = 0.50888893
Iteration 49, loss = 0.50649527
Iteration 50, loss = 0.50664562
Iteration 51, loss = 0.50780019
Iteration 52, loss = 0.50824255
Iteration 53, loss = 0.50728359
Iteration 54, loss = 0.50761642
Iteration 55, loss = 0.50652715
Iteration 56, loss = 0.50814614
Iteration 57, loss = 0.50630885
```

```
Iteration 58, loss = 0.50572652
Iteration 59, loss = 0.50544820
Iteration 60, loss = 0.50467047
Iteration 61, loss = 0.50621356
Iteration 62, loss = 0.50590093
Iteration 63, loss = 0.50457348
Iteration 64, loss = 0.50475402
Iteration 65, loss = 0.50308710
Iteration 66, loss = 0.50277348
Iteration 67, loss = 0.50545467
Iteration 68, loss = 0.50332886
Iteration 69, loss = 0.50370637
Iteration 70, loss = 0.50323303
Iteration 71, loss = 0.50486470
Iteration 72, loss = 0.50293636
Iteration 73, loss = 0.50389007
Iteration 74, loss = 0.50192273
Iteration 75, loss = 0.50399373
Iteration 76, loss = 0.50092663
Iteration 77, loss = 0.50131252
Iteration 78, loss = 0.50103646
Iteration 79, loss = 0.50157618
Iteration 80, loss = 0.50260870
Iteration 81, loss = 0.50235354
Iteration 82, loss = 0.50231471
Iteration 83, loss = 0.50119870
Iteration 84, loss = 0.50153089
Iteration 85, loss = 0.50002778
Iteration 86, loss = 0.50094112
Iteration 87, loss = 0.49993106
Iteration 88, loss = 0.50088958
Iteration 89, loss = 0.49970524
Iteration 90, loss = 0.49925712
Iteration 91, loss = 0.49937781
Iteration 92, loss = 0.49944046
Iteration 93, loss = 0.49914985
Iteration 94, loss = 0.49938230
Iteration 95, loss = 0.49906229
Iteration 96, loss = 0.49903488
Iteration 97, loss = 0.49807852
Iteration 98, loss = 0.49882189
Iteration 99, loss = 0.49844711
Iteration 100, loss = 0.49804937
Iteration 101, loss = 0.49831875
Iteration 102, loss = 0.49911079
Iteration 103, loss = 0.49689668
Iteration 104, loss = 0.49942108
Iteration 105, loss = 0.49719426
Iteration 106, loss = 0.49656599
Iteration 107, loss = 0.49966543
Iteration 108, loss = 0.49562737
Iteration 109, loss = 0.49925062
Iteration 110, loss = 0.49876730
Iteration 111, loss = 0.49690720
Iteration 112, loss = 0.49724468
Iteration 113, loss = 0.49553677
Iteration 114, loss = 0.49529473
```

```
Iteration 115, loss = 0.49731600
Iteration 116, loss = 0.49425996
Iteration 117, loss = 0.49473188
Iteration 118, loss = 0.49215089
Iteration 119, loss = 0.49157343
Iteration 120, loss = 0.49274748
Iteration 121, loss = 0.49286845
Iteration 122, loss = 0.49127318
Iteration 123, loss = 0.48982223
Iteration 124, loss = 0.48863475
Iteration 125, loss = 0.48996216
Iteration 126, loss = 0.48770493
Iteration 127, loss = 0.48521922
Iteration 128, loss = 0.48546529
Iteration 129, loss = 0.48409189
Iteration 130, loss = 0.48299684
Iteration 131, loss = 0.48243170
Iteration 132, loss = 0.47921437
Iteration 133, loss = 0.47758586
Iteration 134, loss = 0.47665852
Iteration 135, loss = 0.47762002
Iteration 136, loss = 0.47582525
Iteration 137, loss = 0.47431663
Iteration 138, loss = 0.47229020
Iteration 139, loss = 0.47135717
Iteration 140, loss = 0.47015680
Iteration 141, loss = 0.47172559
Iteration 142, loss = 0.46677662
Iteration 143, loss = 0.46619842
Iteration 144, loss = 0.46542027
Iteration 145, loss = 0.46340484
Iteration 146, loss = 0.46519102
Iteration 147, loss = 0.46037678
Iteration 148, loss = 0.46129049
Iteration 149, loss = 0.45863680
Iteration 150, loss = 0.45899139
Iteration 151, loss = 0.45690668
Iteration 152, loss = 0.45569829
Iteration 153, loss = 0.45495235
Iteration 154, loss = 0.45446900
Iteration 155, loss = 0.45487381
Iteration 156, loss = 0.45302298
Iteration 157, loss = 0.45321988
Iteration 158, loss = 0.45308177
Iteration 159, loss = 0.45171238
Iteration 160, loss = 0.44948620
Iteration 161, loss = 0.44992587
Iteration 162, loss = 0.45009934
Iteration 163, loss = 0.44863002
Iteration 164, loss = 0.44914111
Iteration 165, loss = 0.45044612
Iteration 166, loss = 0.44921060
Iteration 167, loss = 0.44784701
Iteration 168, loss = 0.44698481
Iteration 169, loss = 0.44648938
Iteration 170, loss = 0.44725572
Iteration 171, loss = 0.44970112
```

```
Iteration 172, loss = 0.44595440
Iteration 173, loss = 0.44507012
Iteration 174, loss = 0.44552067
Iteration 175, loss = 0.44643381
Iteration 176, loss = 0.44585590
Iteration 177, loss = 0.44541969
Iteration 178, loss = 0.44461074
Iteration 179, loss = 0.44460194
Iteration 180, loss = 0.44595715
Iteration 181, loss = 0.44483005
Iteration 182, loss = 0.44528115
Iteration 183, loss = 0.44440955
Iteration 184, loss = 0.44394128
Iteration 185, loss = 0.44538593
Iteration 186, loss = 0.44380366
Iteration 187, loss = 0.44335998
Iteration 188, loss = 0.44380626
Iteration 189, loss = 0.44332811
Iteration 190, loss = 0.44247949
Iteration 191, loss = 0.44301827
Iteration 192, loss = 0.44320955
Iteration 193, loss = 0.44166479
Iteration 194, loss = 0.44533646
Iteration 195, loss = 0.44340913
Iteration 196, loss = 0.44424564
Iteration 197, loss = 0.44166747
Iteration 198, loss = 0.44357133
Iteration 199, loss = 0.44337165
Iteration 200, loss = 0.44074231
Iteration 201, loss = 0.44316341
Iteration 202, loss = 0.44262909
Iteration 203, loss = 0.44258193
Iteration 204, loss = 0.44015327
Iteration 205, loss = 0.44344442
Iteration 206, loss = 0.44224173
Iteration 207, loss = 0.44010040
Iteration 208, loss = 0.44057603
Iteration 209, loss = 0.44243329
Iteration 210, loss = 0.44284133
Iteration 211, loss = 0.44224213
Iteration 212, loss = 0.44074145
Iteration 213, loss = 0.43777729
Iteration 214, loss = 0.44022261
Iteration 215, loss = 0.44144754
Iteration 216, loss = 0.43950021
Iteration 217, loss = 0.43969773
Iteration 218, loss = 0.43957212
Iteration 219, loss = 0.43909787
Iteration 220, loss = 0.44012366
Iteration 221, loss = 0.43842749
Iteration 222, loss = 0.43845760
Iteration 223, loss = 0.44004524
Iteration 224, loss = 0.43790788
Training loss did not improve more than tol=0.000100 for 10 consecutive epoch
s. Stopping.
```

In [51]:
```
preds_nn_omp = nn_omp.predict(X_test_omp)
print(classification_report(y_test, preds_nn_omp))
print(accuracy_score(y_test, preds_nn_omp))
```

```
              precision    recall  f1-score   support

           0       0.80      0.77      0.79     14728
           1       0.70      0.73      0.71     10720

   micro avg       0.76      0.76      0.76     25448
   macro avg       0.75      0.75      0.75     25448
weighted avg       0.76      0.76      0.76     25448

0.7551477522791575
```

In [58]:
```
nn_omp_100 = MLPClassifier(activation='relu', hidden_layer_sizes=(100), verbos
e=2, max_iter=300).fit(X_train_omp, y_train)
```

```
Iteration 1, loss = 0.62793805
Iteration 2, loss = 0.61441455
Iteration 3, loss = 0.61112275
Iteration 4, loss = 0.60848933
Iteration 5, loss = 0.60673952
Iteration 6, loss = 0.60466005
Iteration 7, loss = 0.60242443
Iteration 8, loss = 0.60109354
Iteration 9, loss = 0.59951026
Iteration 10, loss = 0.59781805
Iteration 11, loss = 0.59656726
Iteration 12, loss = 0.59524584
Iteration 13, loss = 0.59375758
Iteration 14, loss = 0.59167808
Iteration 15, loss = 0.59041586
Iteration 16, loss = 0.58852139
Iteration 17, loss = 0.58654496
Iteration 18, loss = 0.58491114
Iteration 19, loss = 0.58230032
Iteration 20, loss = 0.58020912
Iteration 21, loss = 0.57792642
Iteration 22, loss = 0.57511385
Iteration 23, loss = 0.57267962
Iteration 24, loss = 0.56991696
Iteration 25, loss = 0.56683088
Iteration 26, loss = 0.56390343
Iteration 27, loss = 0.56114964
Iteration 28, loss = 0.55828262
Iteration 29, loss = 0.55518457
Iteration 30, loss = 0.55333215
Iteration 31, loss = 0.55015363
Iteration 32, loss = 0.54777815
Iteration 33, loss = 0.54577640
Iteration 34, loss = 0.54291761
Iteration 35, loss = 0.53934540
Iteration 36, loss = 0.53736406
Iteration 37, loss = 0.53557905
Iteration 38, loss = 0.53353672
Iteration 39, loss = 0.53250523
Iteration 40, loss = 0.52929842
Iteration 41, loss = 0.52827765
Iteration 42, loss = 0.52670947
Iteration 43, loss = 0.52528049
Iteration 44, loss = 0.52329127
Iteration 45, loss = 0.52240452
Iteration 46, loss = 0.52068150
Iteration 47, loss = 0.52036542
Iteration 48, loss = 0.51856175
Iteration 49, loss = 0.51882164
Iteration 50, loss = 0.51675975
Iteration 51, loss = 0.51985644
Iteration 52, loss = 0.51868579
Iteration 53, loss = 0.51831862
Iteration 54, loss = 0.51666481
Iteration 55, loss = 0.51665005
Iteration 56, loss = 0.51497381
Iteration 57, loss = 0.51512800
```

```
Iteration 58, loss = 0.51353326
Iteration 59, loss = 0.51401912
Iteration 60, loss = 0.51327753
Iteration 61, loss = 0.51320766
Iteration 62, loss = 0.51253062
Iteration 63, loss = 0.51269105
Iteration 64, loss = 0.51260610
Iteration 65, loss = 0.51110909
Iteration 66, loss = 0.51179933
Iteration 67, loss = 0.51093861
Iteration 68, loss = 0.51150723
Iteration 69, loss = 0.51037942
Iteration 70, loss = 0.51106636
Iteration 71, loss = 0.50998733
Iteration 72, loss = 0.50984413
Iteration 73, loss = 0.51017560
Iteration 74, loss = 0.50997373
Iteration 75, loss = 0.50927877
Iteration 76, loss = 0.50873738
Iteration 77, loss = 0.50842350
Iteration 78, loss = 0.50932578
Iteration 79, loss = 0.50808882
Iteration 80, loss = 0.50908063
Iteration 81, loss = 0.50936931
Iteration 82, loss = 0.50824014
Iteration 83, loss = 0.50878112
Iteration 84, loss = 0.50821620
Iteration 85, loss = 0.50877996
Iteration 86, loss = 0.50875369
Iteration 87, loss = 0.50784806
Iteration 88, loss = 0.50723558
Iteration 89, loss = 0.50844417
Iteration 90, loss = 0.50692944
Iteration 91, loss = 0.50866299
Iteration 92, loss = 0.50748370
Iteration 93, loss = 0.50744612
Iteration 94, loss = 0.50722391
Iteration 95, loss = 0.50671159
Iteration 96, loss = 0.50678972
Iteration 97, loss = 0.50743283
Iteration 98, loss = 0.50604034
Iteration 99, loss = 0.50633217
Iteration 100, loss = 0.50572467
Iteration 101, loss = 0.50586453
Iteration 102, loss = 0.50535146
Iteration 103, loss = 0.50723126
Iteration 104, loss = 0.50624753
Iteration 105, loss = 0.50616155
Iteration 106, loss = 0.50581470
Iteration 107, loss = 0.50601929
Iteration 108, loss = 0.50586694
Iteration 109, loss = 0.50506642
Iteration 110, loss = 0.50556449
Iteration 111, loss = 0.50540333
Iteration 112, loss = 0.50535184
Iteration 113, loss = 0.50545296
Iteration 114, loss = 0.50504111
```

```
Iteration 115, loss = 0.50512596
Iteration 116, loss = 0.50476735
Iteration 117, loss = 0.50481436
Iteration 118, loss = 0.50492044
Iteration 119, loss = 0.50375237
Iteration 120, loss = 0.50486920
Iteration 121, loss = 0.50549932
Iteration 122, loss = 0.50435800
Iteration 123, loss = 0.50409724
Iteration 124, loss = 0.50436441
Iteration 125, loss = 0.50497076
Iteration 126, loss = 0.50335278
Iteration 127, loss = 0.50489882
Iteration 128, loss = 0.50454546
Iteration 129, loss = 0.50409824
Iteration 130, loss = 0.50437083
Iteration 131, loss = 0.50393352
Iteration 132, loss = 0.50442187
Iteration 133, loss = 0.50381280
Iteration 134, loss = 0.50373199
Iteration 135, loss = 0.50384499
Iteration 136, loss = 0.50353212
Iteration 137, loss = 0.50362914
Training loss did not improve more than tol=0.000100 for 10 consecutive epoch
s. Stopping.
```

In [60]:

```
preds_nn_omp_100 = nn_omp_100.predict(X_test_omp)
print(classification_report(y_test, preds_nn_omp_100))
print(accuracy_score(y_test, preds_nn_omp_100))
```

```
              precision    recall  f1-score   support

           0       0.79      0.77      0.78     14728
           1       0.70      0.72      0.71     10720

   micro avg       0.75      0.75      0.75     25448
   macro avg       0.74      0.74      0.74     25448
weighted avg       0.75      0.75      0.75     25448

0.7487032379754794
```

TMP

In [52]:
```
nn_tmp = MLPClassifier(activation='relu', hidden_layer_sizes=(100,80), verbose
=2, max_iter=300).fit(X_train_tmp, y_train)
```

```
Iteration 1, loss = 0.62992168
Iteration 2, loss = 0.62195180
Iteration 3, loss = 0.61952545
Iteration 4, loss = 0.61680725
Iteration 5, loss = 0.61440978
Iteration 6, loss = 0.61325110
Iteration 7, loss = 0.60971068
Iteration 8, loss = 0.60755812
Iteration 9, loss = 0.60405611
Iteration 10, loss = 0.60079038
Iteration 11, loss = 0.59767283
Iteration 12, loss = 0.59377698
Iteration 13, loss = 0.59076214
Iteration 14, loss = 0.58925097
Iteration 15, loss = 0.58623702
Iteration 16, loss = 0.58448439
Iteration 17, loss = 0.58333030
Iteration 18, loss = 0.58167536
Iteration 19, loss = 0.58179046
Iteration 20, loss = 0.58003935
Iteration 21, loss = 0.57815067
Iteration 22, loss = 0.57694327
Iteration 23, loss = 0.57668863
Iteration 24, loss = 0.57559064
Iteration 25, loss = 0.57574409
Iteration 26, loss = 0.57595433
Iteration 27, loss = 0.57424022
Iteration 28, loss = 0.57474405
Iteration 29, loss = 0.57306962
Iteration 30, loss = 0.57401497
Iteration 31, loss = 0.57301358
Iteration 32, loss = 0.57311662
Iteration 33, loss = 0.57152657
Iteration 34, loss = 0.57162852
Iteration 35, loss = 0.57060597
Iteration 36, loss = 0.57037297
Iteration 37, loss = 0.57094181
Iteration 38, loss = 0.57037683
Iteration 39, loss = 0.56910706
Iteration 40, loss = 0.56995332
Iteration 41, loss = 0.56905124
Iteration 42, loss = 0.56759582
Iteration 43, loss = 0.56825889
Iteration 44, loss = 0.56894079
Iteration 45, loss = 0.56816645
Iteration 46, loss = 0.56782782
Iteration 47, loss = 0.56738984
Iteration 48, loss = 0.56715760
Iteration 49, loss = 0.56851194
Iteration 50, loss = 0.56615370
Iteration 51, loss = 0.56639304
Iteration 52, loss = 0.56615832
Iteration 53, loss = 0.56610044
Iteration 54, loss = 0.56559729
Iteration 55, loss = 0.56529838
Iteration 56, loss = 0.56591157
Iteration 57, loss = 0.56543108
```

```
Iteration 58, loss = 0.56520256
Iteration 59, loss = 0.56495937
Iteration 60, loss = 0.56495485
Iteration 61, loss = 0.56459052
Iteration 62, loss = 0.56460016
Iteration 63, loss = 0.56334789
Iteration 64, loss = 0.56427218
Iteration 65, loss = 0.56373665
Iteration 66, loss = 0.56377357
Iteration 67, loss = 0.56257193
Iteration 68, loss = 0.56298122
Iteration 69, loss = 0.56395513
Iteration 70, loss = 0.56209282
Iteration 71, loss = 0.56312054
Iteration 72, loss = 0.56300775
Iteration 73, loss = 0.56249820
Iteration 74, loss = 0.56181596
Iteration 75, loss = 0.56177075
Iteration 76, loss = 0.56101709
Iteration 77, loss = 0.56197075
Iteration 78, loss = 0.56115898
Iteration 79, loss = 0.56128227
Iteration 80, loss = 0.56068092
Iteration 81, loss = 0.56079545
Iteration 82, loss = 0.56073970
Iteration 83, loss = 0.56038326
Iteration 84, loss = 0.56064823
Iteration 85, loss = 0.55993137
Iteration 86, loss = 0.55890065
Iteration 87, loss = 0.55951491
Iteration 88, loss = 0.55988632
Iteration 89, loss = 0.55876039
Iteration 90, loss = 0.56047995
Iteration 91, loss = 0.55862443
Iteration 92, loss = 0.55890445
Iteration 93, loss = 0.55845499
Iteration 94, loss = 0.55834242
Iteration 95, loss = 0.55796022
Iteration 96, loss = 0.55792854
Iteration 97, loss = 0.55728056
Iteration 98, loss = 0.55704316
Iteration 99, loss = 0.55632709
Iteration 100, loss = 0.55630562
Iteration 101, loss = 0.55599914
Iteration 102, loss = 0.55503501
Iteration 103, loss = 0.55463363
Iteration 104, loss = 0.55364585
Iteration 105, loss = 0.55296374
Iteration 106, loss = 0.55234981
Iteration 107, loss = 0.55250045
Iteration 108, loss = 0.55152066
Iteration 109, loss = 0.55086015
Iteration 110, loss = 0.54882424
Iteration 111, loss = 0.54895109
Iteration 112, loss = 0.54796779
Iteration 113, loss = 0.54780958
Iteration 114, loss = 0.54683507
```

```
Iteration 115, loss = 0.54649766
Iteration 116, loss = 0.54482478
Iteration 117, loss = 0.54429071
Iteration 118, loss = 0.54496841
Iteration 119, loss = 0.54315840
Iteration 120, loss = 0.54291216
Iteration 121, loss = 0.54255120
Iteration 122, loss = 0.54272405
Iteration 123, loss = 0.54022810
Iteration 124, loss = 0.54117775
Iteration 125, loss = 0.54000389
Iteration 126, loss = 0.53937999
Iteration 127, loss = 0.53894457
Iteration 128, loss = 0.53935995
Iteration 129, loss = 0.53898134
Iteration 130, loss = 0.53809331
Iteration 131, loss = 0.53838155
Iteration 132, loss = 0.53644947
Iteration 133, loss = 0.53649423
Iteration 134, loss = 0.53728073
Iteration 135, loss = 0.53581707
Iteration 136, loss = 0.53661309
Iteration 137, loss = 0.53689878
Iteration 138, loss = 0.53528891
Iteration 139, loss = 0.53505995
Iteration 140, loss = 0.53479080
Iteration 141, loss = 0.53404663
Iteration 142, loss = 0.53445581
Iteration 143, loss = 0.53498234
Iteration 144, loss = 0.53262080
Iteration 145, loss = 0.53283329
Iteration 146, loss = 0.53168767
Iteration 147, loss = 0.53227551
Iteration 148, loss = 0.53337883
Iteration 149, loss = 0.53188835
Iteration 150, loss = 0.53196357
Iteration 151, loss = 0.53054233
Iteration 152, loss = 0.53169637
Iteration 153, loss = 0.53053104
Iteration 154, loss = 0.53156786
Iteration 155, loss = 0.53116489
Iteration 156, loss = 0.53007373
Iteration 157, loss = 0.53038314
Iteration 158, loss = 0.53148730
Iteration 159, loss = 0.52986153
Iteration 160, loss = 0.52916288
Iteration 161, loss = 0.52833951
Iteration 162, loss = 0.52995896
Iteration 163, loss = 0.53009383
Iteration 164, loss = 0.52709602
Iteration 165, loss = 0.52915678
Iteration 166, loss = 0.52807448
Iteration 167, loss = 0.52761132
Iteration 168, loss = 0.52766418
Iteration 169, loss = 0.52896619
Iteration 170, loss = 0.52829868
Iteration 171, loss = 0.52739682
```

```
Iteration 172, loss = 0.52811302
Iteration 173, loss = 0.52814900
Iteration 174, loss = 0.52810915
Iteration 175, loss = 0.52624454
Iteration 176, loss = 0.52785313
Iteration 177, loss = 0.52708742
Iteration 178, loss = 0.52689253
Iteration 179, loss = 0.52693869
Iteration 180, loss = 0.52547127
Iteration 181, loss = 0.52752757
Iteration 182, loss = 0.52541005
Iteration 183, loss = 0.52458713
Iteration 184, loss = 0.52639868
Iteration 185, loss = 0.52685214
Iteration 186, loss = 0.52506237
Iteration 187, loss = 0.52542009
Iteration 188, loss = 0.52586455
Iteration 189, loss = 0.52542851
Iteration 190, loss = 0.52473072
Iteration 191, loss = 0.52577639
Iteration 192, loss = 0.52425120
Iteration 193, loss = 0.52436393
Iteration 194, loss = 0.52488273
Iteration 195, loss = 0.52405463
Iteration 196, loss = 0.52429712
Iteration 197, loss = 0.52495066
Iteration 198, loss = 0.52483969
Iteration 199, loss = 0.52494605
Iteration 200, loss = 0.52332801
Iteration 201, loss = 0.52475123
Iteration 202, loss = 0.52498535
Iteration 203, loss = 0.52217034
Iteration 204, loss = 0.52244361
Iteration 205, loss = 0.52425435
Iteration 206, loss = 0.52296830
Iteration 207, loss = 0.52439768
Iteration 208, loss = 0.52431602
Iteration 209, loss = 0.52272927
Iteration 210, loss = 0.52181631
Iteration 211, loss = 0.52372949
Iteration 212, loss = 0.52267450
Iteration 213, loss = 0.52283790
Iteration 214, loss = 0.52169743
Iteration 215, loss = 0.52180358
Iteration 216, loss = 0.52299082
Iteration 217, loss = 0.52218255
Iteration 218, loss = 0.52080591
Iteration 219, loss = 0.52136765
Iteration 220, loss = 0.52056965
Iteration 221, loss = 0.52175647
Iteration 222, loss = 0.52039729
Iteration 223, loss = 0.52256753
Iteration 224, loss = 0.52069665
Iteration 225, loss = 0.52106162
Iteration 226, loss = 0.52090785
Iteration 227, loss = 0.51972638
Iteration 228, loss = 0.52100415
```

```
Iteration 229, loss = 0.52146572
Iteration 230, loss = 0.52031869
Iteration 231, loss = 0.52197578
Iteration 232, loss = 0.52016628
Iteration 233, loss = 0.52006401
Iteration 234, loss = 0.52188493
Iteration 235, loss = 0.52045561
Iteration 236, loss = 0.52122998
Iteration 237, loss = 0.52118321
Iteration 238, loss = 0.51897826
Iteration 239, loss = 0.52060763
Iteration 240, loss = 0.51872319
Iteration 241, loss = 0.52072800
Iteration 242, loss = 0.51956823
Iteration 243, loss = 0.51856971
Iteration 244, loss = 0.51779997
Iteration 245, loss = 0.51928293
Iteration 246, loss = 0.51920593
Iteration 247, loss = 0.51795930
Iteration 248, loss = 0.51932803
Iteration 249, loss = 0.51780857
Iteration 250, loss = 0.52033687
Iteration 251, loss = 0.51813666
Iteration 252, loss = 0.51888944
Iteration 253, loss = 0.51889900
Iteration 254, loss = 0.51858738
Iteration 255, loss = 0.51773599
Training loss did not improve more than tol=0.000100 for 10 consecutive epoch
s. Stopping.
```

In [53]:

```
preds_nn_tmp = nn_tmp.predict(X_test_tmp)
print(classification_report(y_test, preds_nn_tmp))
print(accuracy_score(y_test, preds_nn_tmp))
```

```
              precision    recall  f1-score   support

           0       0.78      0.70      0.73     14728
           1       0.64      0.72      0.68     10720

   micro avg       0.71      0.71      0.71     25448
   macro avg       0.71      0.71      0.71     25448
weighted avg       0.72      0.71      0.71     25448

0.7081892486639422
```

SMP

In [54]:
```
nn_smp = MLPClassifier(activation='relu', hidden_layer_sizes=(100,80), verbose
=2, max_iter=300).fit(X_train_smp, y_train)
```

```
Iteration 1, loss = 0.63426709
Iteration 2, loss = 0.62697248
Iteration 3, loss = 0.62521253
Iteration 4, loss = 0.62377549
Iteration 5, loss = 0.62210485
Iteration 6, loss = 0.62094499
Iteration 7, loss = 0.61952160
Iteration 8, loss = 0.61779776
Iteration 9, loss = 0.61658138
Iteration 10, loss = 0.61410614
Iteration 11, loss = 0.61319939
Iteration 12, loss = 0.61089326
Iteration 13, loss = 0.60999489
Iteration 14, loss = 0.60846951
Iteration 15, loss = 0.60792922
Iteration 16, loss = 0.60629016
Iteration 17, loss = 0.60517324
Iteration 18, loss = 0.60452601
Iteration 19, loss = 0.60293621
Iteration 20, loss = 0.60235038
Iteration 21, loss = 0.60180341
Iteration 22, loss = 0.60119721
Iteration 23, loss = 0.60010946
Iteration 24, loss = 0.60018511
Iteration 25, loss = 0.60020483
Iteration 26, loss = 0.59945571
Iteration 27, loss = 0.59890823
Iteration 28, loss = 0.59869935
Iteration 29, loss = 0.59757025
Iteration 30, loss = 0.59701197
Iteration 31, loss = 0.59723442
Iteration 32, loss = 0.59692412
Iteration 33, loss = 0.59695367
Iteration 34, loss = 0.59590619
Iteration 35, loss = 0.59675537
Iteration 36, loss = 0.59638463
Iteration 37, loss = 0.59632023
Iteration 38, loss = 0.59529832
Iteration 39, loss = 0.59544544
Iteration 40, loss = 0.59501439
Iteration 41, loss = 0.59487090
Iteration 42, loss = 0.59469024
Iteration 43, loss = 0.59435526
Iteration 44, loss = 0.59418689
Iteration 45, loss = 0.59368648
Iteration 46, loss = 0.59346989
Iteration 47, loss = 0.59348641
Iteration 48, loss = 0.59316753
Iteration 49, loss = 0.59300349
Iteration 50, loss = 0.59321213
Iteration 51, loss = 0.59271719
Iteration 52, loss = 0.59279914
Iteration 53, loss = 0.59194234
Iteration 54, loss = 0.59241892
Iteration 55, loss = 0.59097166
Iteration 56, loss = 0.59127356
Iteration 57, loss = 0.59127930
```

```
Iteration 58, loss = 0.59079631
Iteration 59, loss = 0.59085278
Iteration 60, loss = 0.59098030
Iteration 61, loss = 0.59017061
Iteration 62, loss = 0.59040138
Iteration 63, loss = 0.58957749
Iteration 64, loss = 0.58952324
Iteration 65, loss = 0.58950862
Iteration 66, loss = 0.58880964
Iteration 67, loss = 0.58814951
Iteration 68, loss = 0.58907710
Iteration 69, loss = 0.58796671
Iteration 70, loss = 0.58744699
Iteration 71, loss = 0.58733306
Iteration 72, loss = 0.58744850
Iteration 73, loss = 0.58731499
Iteration 74, loss = 0.58665859
Iteration 75, loss = 0.58703838
Iteration 76, loss = 0.58637226
Iteration 77, loss = 0.58616341
Iteration 78, loss = 0.58594011
Iteration 79, loss = 0.58548742
Iteration 80, loss = 0.58517609
Iteration 81, loss = 0.58589317
Iteration 82, loss = 0.58500354
Iteration 83, loss = 0.58457233
Iteration 84, loss = 0.58461766
Iteration 85, loss = 0.58461247
Iteration 86, loss = 0.58386111
Iteration 87, loss = 0.58393858
Iteration 88, loss = 0.58322187
Iteration 89, loss = 0.58308064
Iteration 90, loss = 0.58292317
Iteration 91, loss = 0.58236651
Iteration 92, loss = 0.58292517
Iteration 93, loss = 0.58173530
Iteration 94, loss = 0.58187844
Iteration 95, loss = 0.58173967
Iteration 96, loss = 0.58129593
Iteration 97, loss = 0.58084566
Iteration 98, loss = 0.58095169
Iteration 99, loss = 0.58114832
Iteration 100, loss = 0.58014023
Iteration 101, loss = 0.57975607
Iteration 102, loss = 0.57945586
Iteration 103, loss = 0.57944628
Iteration 104, loss = 0.57919119
Iteration 105, loss = 0.57950931
Iteration 106, loss = 0.57864712
Iteration 107, loss = 0.57824056
Iteration 108, loss = 0.57907375
Iteration 109, loss = 0.57851291
Iteration 110, loss = 0.57840137
Iteration 111, loss = 0.57800658
Iteration 112, loss = 0.57714378
Iteration 113, loss = 0.57757163
Iteration 114, loss = 0.57636273
```

```
Iteration 115, loss = 0.57672675
Iteration 116, loss = 0.57676003
Iteration 117, loss = 0.57664588
Iteration 118, loss = 0.57574430
Iteration 119, loss = 0.57611488
Iteration 120, loss = 0.57577676
Iteration 121, loss = 0.57558774
Iteration 122, loss = 0.57506838
Iteration 123, loss = 0.57544133
Iteration 124, loss = 0.57516735
Iteration 125, loss = 0.57438556
Iteration 126, loss = 0.57347553
Iteration 127, loss = 0.57430542
Iteration 128, loss = 0.57411756
Iteration 129, loss = 0.57363839
Iteration 130, loss = 0.57369372
Iteration 131, loss = 0.57448377
Iteration 132, loss = 0.57397660
Iteration 133, loss = 0.57225573
Iteration 134, loss = 0.57279493
Iteration 135, loss = 0.57194108
Iteration 136, loss = 0.57228615
Iteration 137, loss = 0.57259797
Iteration 138, loss = 0.57277657
Iteration 139, loss = 0.57131707
Iteration 140, loss = 0.57188325
Iteration 141, loss = 0.57180194
Iteration 142, loss = 0.57083019
Iteration 143, loss = 0.57037574
Iteration 144, loss = 0.57061882
Iteration 145, loss = 0.57146403
Iteration 146, loss = 0.57076126
Iteration 147, loss = 0.57131323
Iteration 148, loss = 0.56953186
Iteration 149, loss = 0.57062247
Iteration 150, loss = 0.56985404
Iteration 151, loss = 0.57029659
Iteration 152, loss = 0.56886065
Iteration 153, loss = 0.56936772
Iteration 154, loss = 0.56974665
Iteration 155, loss = 0.56901688
Iteration 156, loss = 0.56866889
Iteration 157, loss = 0.56819342
Iteration 158, loss = 0.56796714
Iteration 159, loss = 0.56792580
Iteration 160, loss = 0.56753179
Iteration 161, loss = 0.56743443
Iteration 162, loss = 0.56744616
Iteration 163, loss = 0.56684169
Iteration 164, loss = 0.56767913
Iteration 165, loss = 0.56746590
Iteration 166, loss = 0.56812406
Iteration 167, loss = 0.56722870
Iteration 168, loss = 0.56696287
Iteration 169, loss = 0.56600087
Iteration 170, loss = 0.56648749
Iteration 171, loss = 0.56619309
```

```
Iteration 172, loss = 0.56574852
Iteration 173, loss = 0.56639718
Iteration 174, loss = 0.56548959
Iteration 175, loss = 0.56619337
Iteration 176, loss = 0.56521621
Iteration 177, loss = 0.56543438
Iteration 178, loss = 0.56454989
Iteration 179, loss = 0.56548920
Iteration 180, loss = 0.56481017
Iteration 181, loss = 0.56615458
Iteration 182, loss = 0.56595163
Iteration 183, loss = 0.56460522
Iteration 184, loss = 0.56467701
Iteration 185, loss = 0.56480382
Iteration 186, loss = 0.56468487
Iteration 187, loss = 0.56476562
Iteration 188, loss = 0.56436392
Iteration 189, loss = 0.56433853
Iteration 190, loss = 0.56424671
Iteration 191, loss = 0.56412377
Iteration 192, loss = 0.56394953
Iteration 193, loss = 0.56354585
Iteration 194, loss = 0.56326457
Iteration 195, loss = 0.56405301
Iteration 196, loss = 0.56219697
Iteration 197, loss = 0.56220989
Iteration 198, loss = 0.56202644
Iteration 199, loss = 0.56285979
Iteration 200, loss = 0.56347660
Iteration 201, loss = 0.56146176
Iteration 202, loss = 0.56273221
Iteration 203, loss = 0.56152994
Iteration 204, loss = 0.56208292
Iteration 205, loss = 0.56305875
Iteration 206, loss = 0.56152097
Iteration 207, loss = 0.56215980
Iteration 208, loss = 0.56121828
Iteration 209, loss = 0.56227051
Iteration 210, loss = 0.56287290
Iteration 211, loss = 0.56187803
Iteration 212, loss = 0.56066086
Iteration 213, loss = 0.56061060
Iteration 214, loss = 0.56123330
Iteration 215, loss = 0.56146490
Iteration 216, loss = 0.56086668
Iteration 217, loss = 0.56057285
Iteration 218, loss = 0.56033939
Iteration 219, loss = 0.56073222
Iteration 220, loss = 0.56011588
Iteration 221, loss = 0.55969510
Iteration 222, loss = 0.55947331
Iteration 223, loss = 0.56019943
Iteration 224, loss = 0.55989442
Iteration 225, loss = 0.55921947
Iteration 226, loss = 0.55975060
Iteration 227, loss = 0.55995482
Iteration 228, loss = 0.55874605
```

```
Iteration 229, loss = 0.55858545
Iteration 230, loss = 0.55848216
Iteration 231, loss = 0.55940831
Iteration 232, loss = 0.55869230
Iteration 233, loss = 0.55933326
Iteration 234, loss = 0.55826178
Iteration 235, loss = 0.55929760
Iteration 236, loss = 0.55890180
Iteration 237, loss = 0.55944464
Iteration 238, loss = 0.55841235
Iteration 239, loss = 0.55827163
Iteration 240, loss = 0.55744395
Iteration 241, loss = 0.55902549
Iteration 242, loss = 0.55890618
Iteration 243, loss = 0.55903786
Iteration 244, loss = 0.55706987
Iteration 245, loss = 0.55833982
Iteration 246, loss = 0.55704232
Iteration 247, loss = 0.55673554
Iteration 248, loss = 0.55752136
Iteration 249, loss = 0.55795159
Iteration 250, loss = 0.55735108
Iteration 251, loss = 0.55757929
Iteration 252, loss = 0.55683345
Iteration 253, loss = 0.55684328
Iteration 254, loss = 0.55742812
Iteration 255, loss = 0.55617373
Iteration 256, loss = 0.55538584
Iteration 257, loss = 0.55736830
Iteration 258, loss = 0.55720018
Iteration 259, loss = 0.55595150
Iteration 260, loss = 0.55573225
Iteration 261, loss = 0.55763511
Iteration 262, loss = 0.55554535
Iteration 263, loss = 0.55599716
Iteration 264, loss = 0.55525252
Iteration 265, loss = 0.55549041
Iteration 266, loss = 0.55564399
Iteration 267, loss = 0.55676007
Iteration 268, loss = 0.55592097
Iteration 269, loss = 0.55461068
Iteration 270, loss = 0.55509326
Iteration 271, loss = 0.55478505
Iteration 272, loss = 0.55486606
Iteration 273, loss = 0.55430845
Iteration 274, loss = 0.55496763
Iteration 275, loss = 0.55539981
Iteration 276, loss = 0.55475109
Iteration 277, loss = 0.55442127
Iteration 278, loss = 0.55478982
Iteration 279, loss = 0.55529607
Iteration 280, loss = 0.55412862
Iteration 281, loss = 0.55478859
Iteration 282, loss = 0.55441710
Iteration 283, loss = 0.55445984
Iteration 284, loss = 0.55301805
Iteration 285, loss = 0.55373276
```

```
Iteration 286, loss = 0.55490921
Iteration 287, loss = 0.55469303
Iteration 288, loss = 0.55519017
Iteration 289, loss = 0.55289520
Iteration 290, loss = 0.55382844
Iteration 291, loss = 0.55382434
Iteration 292, loss = 0.55323021
Iteration 293, loss = 0.55332220
Iteration 294, loss = 0.55376254
Iteration 295, loss = 0.55553203
Iteration 296, loss = 0.55420526
Iteration 297, loss = 0.55328592
Iteration 298, loss = 0.55378531
Iteration 299, loss = 0.55244946
Iteration 300, loss = 0.55282801
```

In [55]:

```
preds_nn_smp = nn_smp.predict(X_test_smp)
print(classification_report(y_test, preds_nn_smp))
print(accuracy_score(y_test, preds_nn_smp))
```

```
              precision    recall  f1-score   support

           0       0.70      0.77      0.73     14728
           1       0.63      0.55      0.58     10720

   micro avg       0.67      0.67      0.67     25448
   macro avg       0.66      0.66      0.66     25448
weighted avg       0.67      0.67      0.67     25448

0.6730980823640365
```

In [ ]:

RANDOM FOREST

In [63]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [62]:

```
params = {'max_depth':[8,10,12,14,16]}
```

OMP

In [65]:
```
rf_omp = GridSearchCV(RandomForestClassifier(n_jobs=-1), params, verbose=2, cv
=5, n_jobs=-1).fit(X_train_omp, y_train)
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  15 out of  25 | elapsed:    8.6s remaining:
5.7s
[Parallel(n_jobs=-1)]: Done  25 out of  25 | elapsed:    9.6s finished
```

In [69]:
```
print(rf_omp.best_estimator_)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=-1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [70]:
```
preds_rf_omp = rf_omp.predict(X_test_omp)
print(classification_report(y_test, preds_rf_omp))
print(accuracy_score(y_test, preds_rf_omp))
```

```
              precision    recall  f1-score   support

           0       0.83      0.75      0.79     14728
           1       0.70      0.79      0.74     10720

   micro avg       0.77      0.77      0.77     25448
   macro avg       0.76      0.77      0.76     25448
weighted avg       0.78      0.77      0.77     25448

0.7671329770512417
```

TMP

In [71]:
```
rf_tmp = GridSearchCV(RandomForestClassifier(n_jobs=-1), params, verbose=2, cv
=5, n_jobs=-1).fit(X_train_tmp, y_train)
print(rf_tmp.best_estimator_)
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  15 out of  25 | elapsed:    5.4s remaining:
3.6s
[Parallel(n_jobs=-1)]: Done  25 out of  25 | elapsed:    7.2s finished

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=14, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=-1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [72]:
```
preds_rf_tmp = rf_tmp.predict(X_test_tmp)
print(classification_report(y_test, preds_rf_tmp))
print(accuracy_score(y_test, preds_rf_tmp))
```

```
              precision    recall  f1-score   support

           0       0.76      0.75      0.75     14728
           1       0.66      0.67      0.67     10720

   micro avg       0.72      0.72      0.72     25448
   macro avg       0.71      0.71      0.71     25448
weighted avg       0.72      0.72      0.72     25448

0.7174237661112858
```

SMP

In [73]:
```
rf_smp = GridSearchCV(RandomForestClassifier(n_jobs=-1), params, verbose=2, cv
=5, n_jobs=-1).fit(X_train_smp, y_train)
print(rf_smp.best_estimator_)
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  15 out of  25 | elapsed:    8.5s remaining:
5.6s
[Parallel(n_jobs=-1)]: Done  25 out of  25 | elapsed:   10.9s finished

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=12, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=-1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [74]:
```
preds_rf_smp = rf_smp.predict(X_test_smp)
print(classification_report(y_test, preds_rf_smp))
print(accuracy_score(y_test, preds_rf_smp))
```

```
              precision    recall  f1-score   support

           0       0.71      0.76      0.74     14728
           1       0.64      0.58      0.61     10720

   micro avg       0.68      0.68      0.68     25448
   macro avg       0.68      0.67      0.67     25448
weighted avg       0.68      0.68      0.68     25448

0.6849261238604213
```

In [ ]:

REPORTS

In [75]:
```
probs_lr_omp = lr_omp.predict_proba(X_test_omp)[:,1]
probs_lr_tmp = lr_tmp.predict_proba(X_test_tmp)[:,1]
probs_lr_smp = lr_smp.predict_proba(X_test_smp)[:,1]
probs_qda_omp = qda_omp.predict_proba(X_test_omp)[:,1]
probs_qda_tmp = qda_tmp.predict_proba(X_test_tmp)[:,1]
probs_qda_smp = qda_smp.predict_proba(X_test_smp)[:,1]
probs_nn_omp = nn_omp.predict_proba(X_test_omp)[:,1]
probs_nn_tmp = nn_tmp.predict_proba(X_test_tmp)[:,1]
probs_nn_smp = nn_smp.predict_proba(X_test_smp)[:,1]
probs_rf_omp = rf_omp.predict_proba(X_test_omp)[:,1]
probs_rf_tmp = rf_tmp.predict_proba(X_test_tmp)[:,1]
probs_rf_smp = rf_smp.predict_proba(X_test_smp)[:,1]
```

In [76]:
```
d = {'y': y_test, 'lr_omp': probs_lr_omp, 'lr_tmp': probs_lr_tmp, 'lr_smp': pr
obs_lr_smp, \
    'qda_omp': probs_qda_omp, 'qda_tmp': probs_qda_tmp, 'qda_smp': probs_qda_s
mp, \
    'nn_omp': probs_nn_omp, 'nn_tmp': probs_nn_tmp, 'nn_smp': probs_nn_smp, \
    'rf_omp': probs_rf_omp, 'rf_tmp': probs_rf_tmp, 'rf_smp': probs_rf_smp}
results = pd.DataFrame(data=d)
```

In [77]:
```
results.to_csv('./data/model_probs.csv', sep=',', header=True)
```

In [94]:
```
ids = X_test.index
test_obs = df_filtered.ix[ids]
```

In [95]:
```
test_obs['Predicted'] = pd.Series(np.array(y_test), index=test_obs.index)
```

In [96]:
```
test_obs['LR OMP PRED'] = pd.Series(np.array(preds_lr_omp), index=test_obs.ind
ex)
test_obs['LR TMP PRED'] = pd.Series(np.array(preds_lr_tmp), index=test_obs.ind
ex)
test_obs['LR SMP PRED'] = pd.Series(np.array(preds_lr_smp), index=test_obs.ind
ex)
test_obs['QDA OMP PRED'] = pd.Series(np.array(preds_qda_omp), index=test_obs.i
ndex)
test_obs['QDA TMP PRED'] = pd.Series(np.array(preds_qda_tmp), index=test_obs.i
ndex)
test_obs['QDA SMP PRED'] = pd.Series(np.array(preds_qda_smp), index=test_obs.i
ndex)
test_obs['NN OMP PRED'] = pd.Series(np.array(preds_nn_omp), index=test_obs.ind
ex)
test_obs['NN TMP PRED'] = pd.Series(np.array(preds_nn_tmp), index=test_obs.ind
ex)
test_obs['NN SMP PRED'] = pd.Series(np.array(preds_nn_smp), index=test_obs.ind
ex)
```

In [97]:
```
test_obs['LR OMP PROB'] = pd.Series(np.array(probs_lr_omp), index=test_obs.ind
ex)
test_obs['LR TMP PROB'] = pd.Series(np.array(probs_lr_tmp), index=test_obs.ind
ex)
test_obs['LR SMP PROB'] = pd.Series(np.array(probs_lr_smp), index=test_obs.ind
ex)
test_obs['QDA OMP PROB'] = pd.Series(np.array(probs_qda_omp), index=test_obs.i
ndex)
test_obs['QDA TMP PROB'] = pd.Series(np.array(probs_qda_tmp), index=test_obs.i
ndex)
test_obs['QDA SMP PROB'] = pd.Series(np.array(probs_qda_smp), index=test_obs.i
ndex)
test_obs['NN OMP PROB'] = pd.Series(np.array(probs_nn_omp), index=test_obs.ind
ex)
test_obs['NN TMP PROB'] = pd.Series(np.array(probs_nn_tmp), index=test_obs.ind
ex)
test_obs['NN SMP PROB'] = pd.Series(np.array(probs_nn_smp), index=test_obs.ind
ex)
```

In [98]:
```
writer = pd.ExcelWriter('./data/final_results.xlsx')
test_obs.to_excel(writer,'Sheet1')
writer.save()
```

In [ ]:
```
```

LASSO

OMP

In [139]:
```
from sklearn.linear_model import LassoCV

lasso_omp = LassoCV(alphas = [0.0005, 0.005, 0.05, 0.5], n_jobs=4).fit(X_train_omp, y_train)

coef = pd.Series(lasso_omp.coef_, index = X_train_omp.columns)
print("Lasso picked " + str(sum(coef != 0)) + " variables and eliminated the other " +
      str(sum(coef == 0)) + " variables")
```

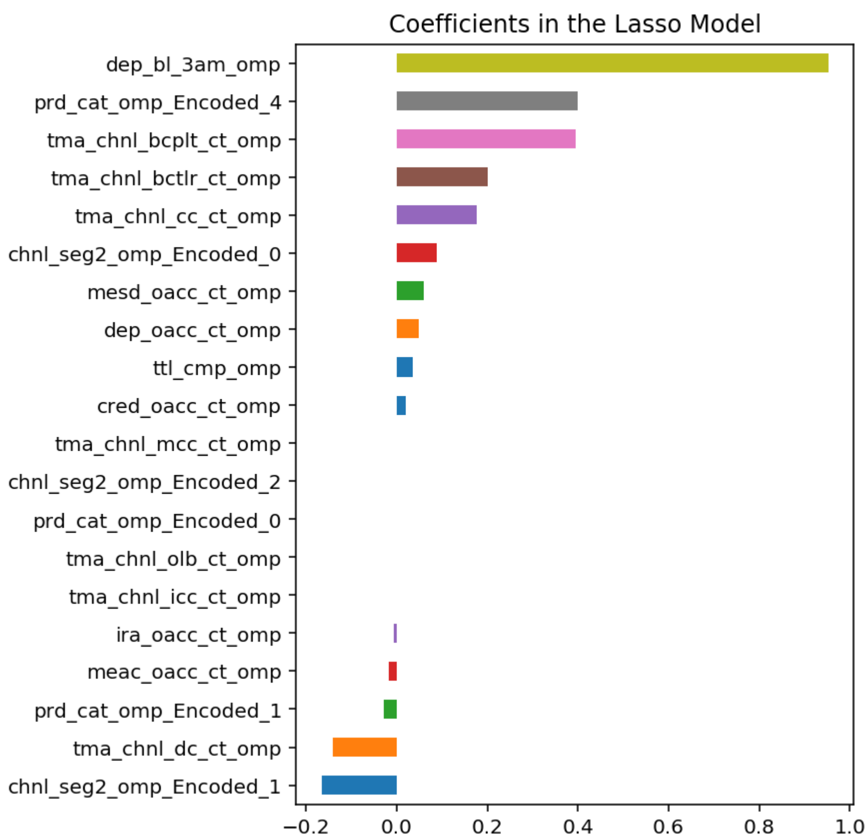
```
Lasso picked 19 variables and eliminated the other 14 variables
```

In [140]:
```
coef.sort_values()
```

Out[140]:
```
chnl_seg2_omp_Encoded_1   -0.166513
tma_chnl_dc_ct_omp        -0.142149
prd_cat_omp_Encoded_1     -0.028834
meac_oacc_ct_omp          -0.018286
ira_oacc_ct_omp           -0.005972
tma_chnl_icc_ct_omp       -0.000611
tma_chnl_olb_ct_omp        0.000000
prd_cat_omp_Encoded_0      0.000000
chnl_seg2_omp_Encoded_2   -0.000000
tma_chnl_mcc_ct_omp        0.000000
tma_chnl_ach_ct_omp        0.000000
tma_chnl_mob_ct_omp        0.000000
prd_cat_omp_Encoded_3      0.000000
tma_chnl_atm_ct_omp        0.000000
prd_cat_omp_Encoded_2     -0.000000
inv_bl_3am_omp             0.000000
cr_bl_3am_omp             -0.000000
fsvc_oacc_ct_omp          -0.000000
oprd_bl_3am_omp            0.000000
inv_oacc_ct_omp           -0.000000
tma_chnl_ccc_ct_omp        0.001719
tma_chnl_dcc_ct_omp        0.004037
chnl_seg2_omp_Encoded_3    0.011306
cred_oacc_ct_omp           0.020126
ttl_cmp_omp                0.036075
dep_oacc_ct_omp            0.049486
mesd_oacc_ct_omp           0.059780
chnl_seg2_omp_Encoded_0    0.088839
tma_chnl_cc_ct_omp         0.177016
tma_chnl_bctlr_ct_omp      0.199779
tma_chnl_bcplt_ct_omp      0.395319
prd_cat_omp_Encoded_4      0.399796
dep_bl_3am_omp             0.952627
dtype: float64
```

In [141]:
```
import matplotlib
import matplotlib.pyplot as plt
%config InlineBackend.figure_format = 'retina'
%matplotlib inline

imp_coef = pd.concat([coef.sort_values().head(10),
                     coef.sort_values().tail(10)])
matplotlib.rcParams['figure.figsize'] = (5, 7)
imp_coef.plot(kind = "barh")
a = plt.title("Coefficients in the Lasso Model")
```



TMP

In [142]:
```
lasso_tmp = LassoCV(alphas = [0.0005, 0.005, 0.05, 0.5], n_jobs=4).fit(X_train
_tmp, y_train)

coef = pd.Series(lasso_tmp.coef_, index = X_train_tmp.columns)
print("Lasso picked " + str(sum(coef != 0)) + " variables and eliminated the o
ther " +
        str(sum(coef == 0)) + " variables")
```

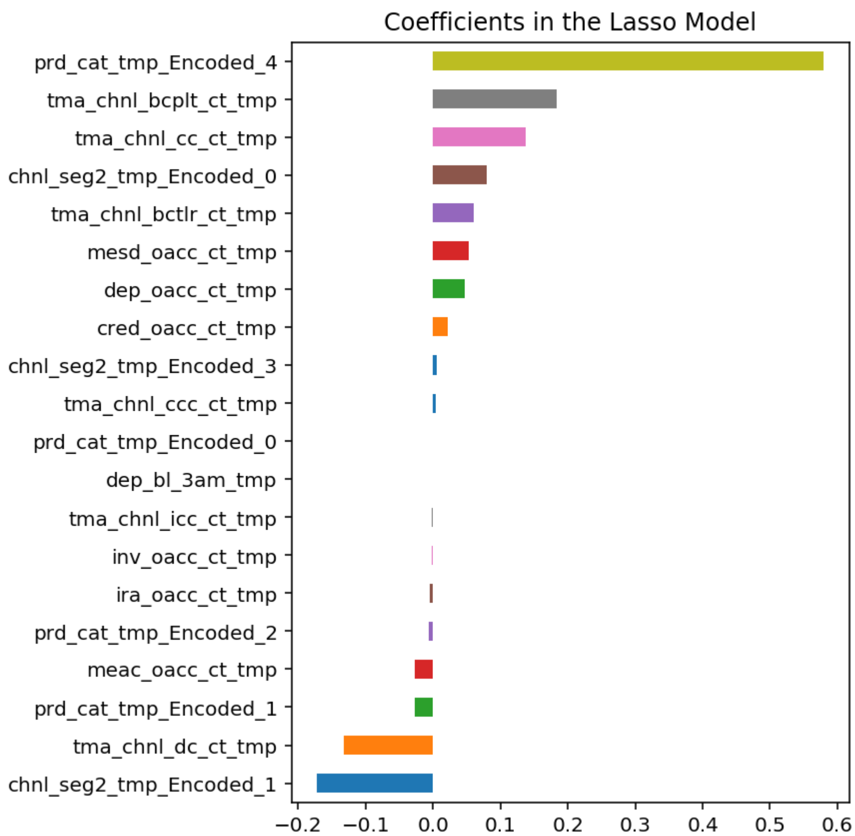
```
Lasso picked 19 variables and eliminated the other 14 variables
```

In [143]:
```
coef.sort_values()
```

Out[143]:
```
chnl_seg2_tmp_Encoded_1   -0.171989
tma_chnl_dc_ct_tmp        -0.132383
prd_cat_tmp_Encoded_1     -0.027306
meac_oacc_ct_tmp          -0.026518
prd_cat_tmp_Encoded_2     -0.006065
ira_oacc_ct_tmp           -0.005170
inv_oacc_ct_tmp           -0.001561
tma_chnl_icc_ct_tmp       -0.001466
dep_bl_3am_tmp             0.000000
prd_cat_tmp_Encoded_0      0.000000
chnl_seg2_tmp_Encoded_2   -0.000000
ttl_cmp_tmp                0.000000
tma_chnl_mcc_ct_tmp        0.000000
tma_chnl_ach_ct_tmp        0.000000
tma_chnl_mob_ct_tmp        0.000000
prd_cat_tmp_Encoded_3      0.000000
tma_chnl_olb_ct_tmp        0.000000
fsvc_oacc_ct_tmp          -0.000000
oprd_bl_3am_tmp            0.000000
cr_bl_3am_tmp             -0.000000
inv_bl_3am_tmp             0.000000
tma_chnl_atm_ct_tmp        0.000000
tma_chnl_dcc_ct_tmp        0.004586
tma_chnl_ccc_ct_tmp        0.004864
chnl_seg2_tmp_Encoded_3    0.005024
cred_oacc_ct_tmp           0.021985
dep_oacc_ct_tmp            0.047206
mesd_oacc_ct_tmp           0.053673
tma_chnl_bctlr_ct_tmp      0.061233
chnl_seg2_tmp_Encoded_0    0.079791
tma_chnl_cc_ct_tmp         0.137852
tma_chnl_bcplt_ct_tmp      0.184247
prd_cat_tmp_Encoded_4      0.579955
dtype: float64
```

In [144]:
```
import matplotlib
import matplotlib.pyplot as plt
%config InlineBackend.figure_format = 'retina'
%matplotlib inline

imp_coef = pd.concat([coef.sort_values().head(10),
                     coef.sort_values().tail(10)])
matplotlib.rcParams['figure.figsize'] = (5, 7)
imp_coef.plot(kind = "barh")
a = plt.title("Coefficients in the Lasso Model")
```



SMP

In [145]:
```
lasso_smp = LassoCV(alphas = [0.0005, 0.005, 0.05, 0.5], n_jobs=4).fit(X_train
_smp, y_train)

coef = pd.Series(lasso_smp.coef_, index = X_train_smp.columns)
print("Lasso picked " + str(sum(coef != 0)) + " variables and eliminated the o
ther " +
        str(sum(coef == 0)) + " variables")
```

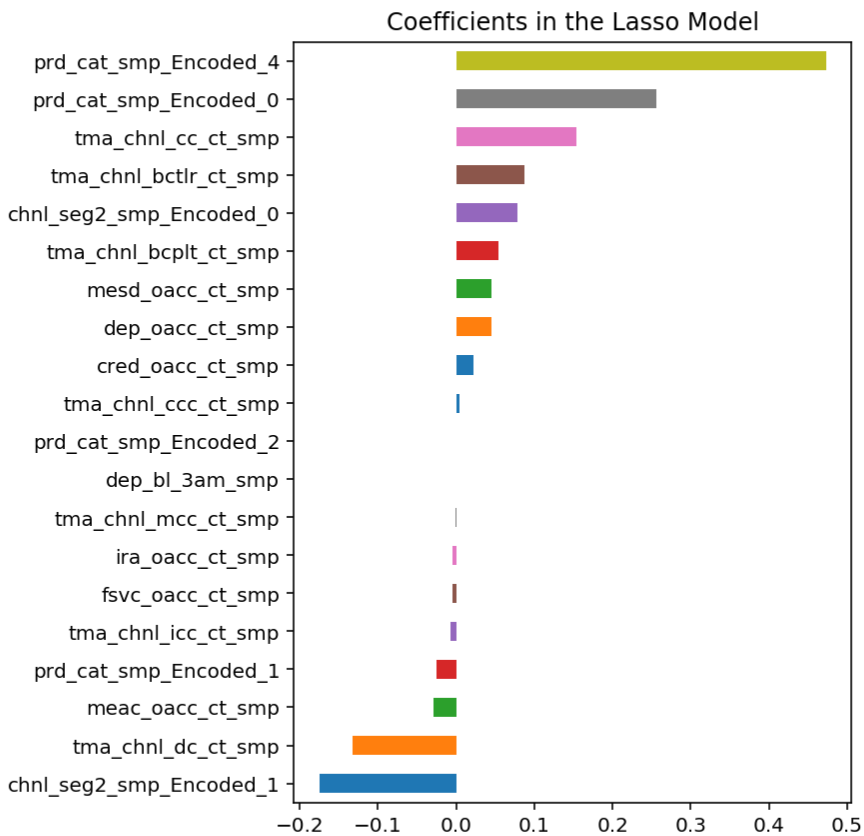
```
Lasso picked 20 variables and eliminated the other 13 variables
```

In [146]:
```
coef.sort_values()
```

Out[146]:
```
chnl_seg2_smp_Encoded_1   -0.175077
tma_chnl_dc_ct_smp        -0.132502
meac_oacc_ct_smp          -0.028402
prd_cat_smp_Encoded_1     -0.025093
tma_chnl_icc_ct_smp       -0.007184
fsvc_oacc_ct_smp          -0.004513
ira_oacc_ct_smp           -0.004427
tma_chnl_mcc_ct_smp       -0.001021
dep_bl_3am_smp             0.000000
prd_cat_smp_Encoded_2     -0.000000
chnl_seg2_smp_Encoded_2   -0.000000
ttl_cmp_smp               -0.000000
tma_chnl_ach_ct_smp        0.000000
tma_chnl_mob_ct_smp        0.000000
prd_cat_smp_Encoded_3     -0.000000
tma_chnl_olb_ct_smp        0.000000
inv_bl_3am_smp             0.000000
inv_oacc_ct_smp           -0.000000
tma_chnl_atm_ct_smp        0.000000
oprd_bl_3am_smp            0.000000
cr_bl_3am_smp             -0.000000
chnl_seg2_smp_Encoded_3    0.002685
tma_chnl_dcc_ct_smp        0.003958
tma_chnl_ccc_ct_smp        0.004670
cred_oacc_ct_smp           0.022187
dep_oacc_ct_smp            0.044556
mesd_oacc_ct_smp           0.045023
tma_chnl_bcplt_ct_smp      0.054423
chnl_seg2_smp_Encoded_0    0.078694
tma_chnl_bctlr_ct_smp      0.087577
tma_chnl_cc_ct_smp         0.154020
prd_cat_smp_Encoded_0      0.255746
prd_cat_smp_Encoded_4      0.473084
dtype: float64
```

In [147]:
```
import matplotlib
import matplotlib.pyplot as plt
%config InlineBackend.figure_format = 'retina'
%matplotlib inline

imp_coef = pd.concat([coef.sort_values().head(10),
                     coef.sort_values().tail(10)])
matplotlib.rcParams['figure.figsize'] = (5, 7)
imp_coef.plot(kind = "barh")
a = plt.title("Coefficients in the Lasso Model")
```



In [ ]:

In [120]:
```
pd.Series(y_test).value_counts(normalize=True)[1]
```

Out[120]: 0.42250864508016345

In [ ]: