modelling_blend

September 23, 2019

[26]:
```python
import pickle

%matplotlib inline
import matplotlib.pyplot as plt

import numpy as np

import pandas as pd
pd.options.display.max_columns = None

from sklearn.model_selection import StratifiedKFold, learning_curve,␣
 ↪train_test_split
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪precision_score, recall_score

from lightgbm import LGBMClassifier

from imblearn.over_sampling import SMOTE
```

```
Using TensorFlow backend.
```

[30]:
```python
# Load preprocessed dataset

PREPROCESSED_DF_PATH = 'preprocessed_applications_df.dataframe.pd'
main_df = pickle.load( open(PREPROCESSED_DF_PATH, 'rb') )
```

[31]:
```python
# Remove redundant features

target_ids = main_df['appl_id']
target_labels = main_df['df']

features_to_drop = [
    'appl_id', 'client_id',
    'app_crtime', 'birth', 'pass_bdate', 'lived_since',␣
 ↪'is_same_reg_lived_since', 'jobsworksince',
    'df'  # target feature
]
```

```
main_df = main_df.drop( features_to_drop, axis=1 )
```

[32]:
```
# Encode several features

features_to_encode = [
    'gender',
    'top_browser', 'top_platform',
    'total_devices_cnt',
    'visit_top_dayofweek', 'visit_top_dayhour',
    'binned_visit_top_dayhour',
    'binned_fam_status', 'binned_quantity_child', 'binned_max_age_child',
    'binned_property',
    'binned_region', 'binned_region_reg',
    'binned_work_experience', 'binned_empl_state', 'binned_empl_type',␣
 ↪'binned_empl_worker_count',
    'binned_education_area', 'binned_education',
    'binned_days_from_birth', 'binned_days_from_passbdate',
    'app_month_num'
]

for feature_name in features_to_encode:
    main_df[feature_name] = main_df[feature_name].astype('category')

main_encoded_df = pd.get_dummies(
    main_df, columns=features_to_encode, drop_first=True
)
```

[33]:
```
def draw_learning_curve(estimator, X_tr, y_tr):
    train_sizes, train_scores, val_scores = learning_curve(
        estimator, X_tr, y_tr, train_sizes=np.linspace(0.1, 1.0, 5), cv=3
    )
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    val_scores_mean = np.mean(val_scores, axis=1)
    val_scores_std = np.std(val_scores, axis=1)
    plt.grid()
    plt.fill_between(
        train_sizes,
        train_scores_mean - train_scores_std,
        train_scores_mean + train_scores_std,
        alpha=0.1, color="r"
    )
    plt.fill_between(
        train_sizes,
        val_scores_mean - val_scores_std,
        val_scores_mean + val_scores_std,
        alpha=0.1, color="g"
    )
```

```
    plt.plot(
        train_sizes,
        train_scores_mean,
        'o-', color="r", label="Training score"
    )
    plt.plot(
        train_sizes,
        val_scores_mean,
        'o-', color="g", label="Cross-validation score"
    )
    plt.legend(loc="best")
    plt.show()
```

[34]:
```
# Prepare train df and train target labels (no upsampling)

train_labels = target_labels[ target_labels.isnull() == False ]
train_df = main_encoded_df.loc[train_labels.index, :]

enc_train_labels = train_labels.map({
    'bad': 0, 'good': 1
})
```

[12]:
```
def try_model( data_df, target_df ):
    skf = StratifiedKFold(
        n_splits=4,
        shuffle=True
    )

    idx = 0
    for train_idx, val_idx in skf.split(data_df, target_df):
        X_tr, X_val = data_df.iloc[train_idx, :], data_df.iloc[val_idx, :]
        y_tr, y_val = target_df.iloc[train_idx], target_df.iloc[val_idx]

        idx += 1
        display('fold {0}/{1}'.format( idx, skf.get_n_splits() ))

        lgbm_model = LGBMClassifier(
            metric='recall',
            objective='binary',
            n_estimators=50,
            learning_rate=0.05,
            scale_pos_weight=1,
            n_jobs=4,
        )
        lgbm_model.fit(X_tr, y_tr)

        val_y_pred = lgbm_model.predict(X_val)
        print( 'accuracy', lgbm_model.score(X_val, y_val) )
```

```
        print( 'recall', recall_score(y_val, val_y_pred ) )
        print( 'precision', precision_score(y_val, val_y_pred) )
        print( '\nconfusion matrix:\n', confusion_matrix(y_val, val_y_pred) )
        print( classification_report(y_val, val_y_pred) )

        draw_learning_curve( lgbm_model, X_tr, y_tr )
```

[13]:

```
# Try out lgbm classifier without upsampling on StratifiedFold

# Results - too bad

try_model(
    train_df, enc_train_labels
)
```

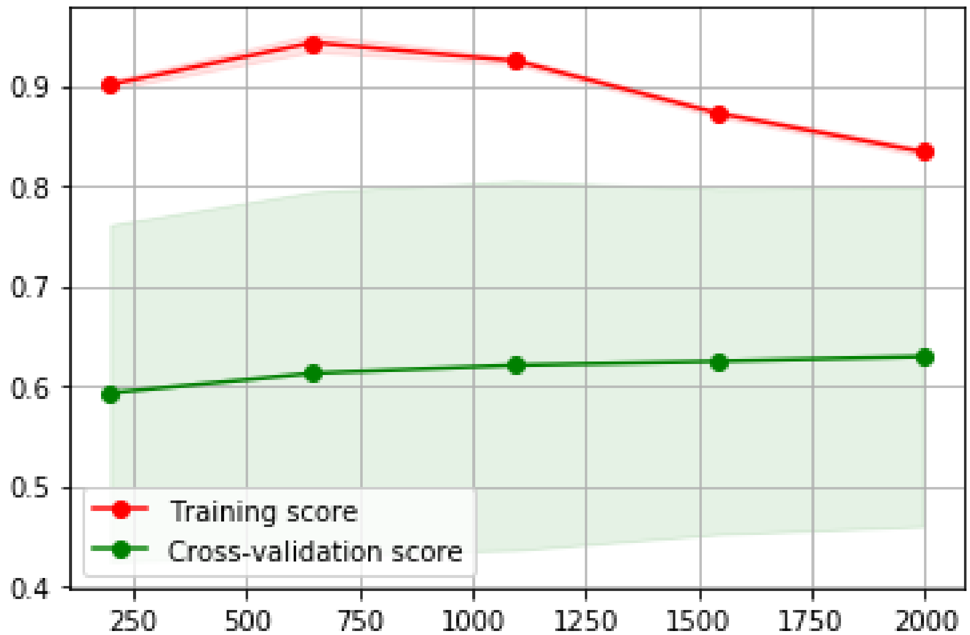
```
'fold 1/4'


accuracy 0.749
recall 0.9867021276595744
precision 0.7548321464903357

confusion matrix:
 [[  7 241]
 [ 10 742]]
              precision    recall  f1-score   support

           0       0.41      0.03      0.05       248
           1       0.75      0.99      0.86       752

    accuracy                           0.75      1000
   macro avg       0.58      0.51      0.45      1000
weighted avg       0.67      0.75      0.66      1000
```



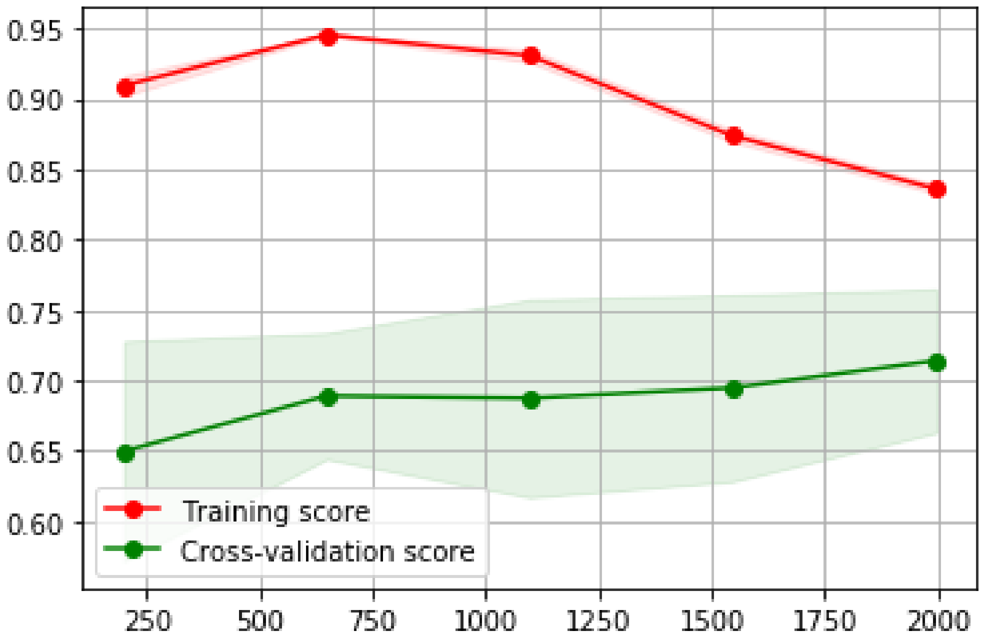

```
'fold 2/4'


accuracy 0.758
recall 0.9920212765957447
precision 0.7596741344195519

confusion matrix:
 [[ 12 236]
 [  6 746]]
              precision    recall  f1-score   support

           0       0.67      0.05      0.09       248
           1       0.76      0.99      0.86       752

    accuracy                           0.76      1000
   macro avg       0.71      0.52      0.48      1000
weighted avg       0.74      0.76      0.67      1000
```



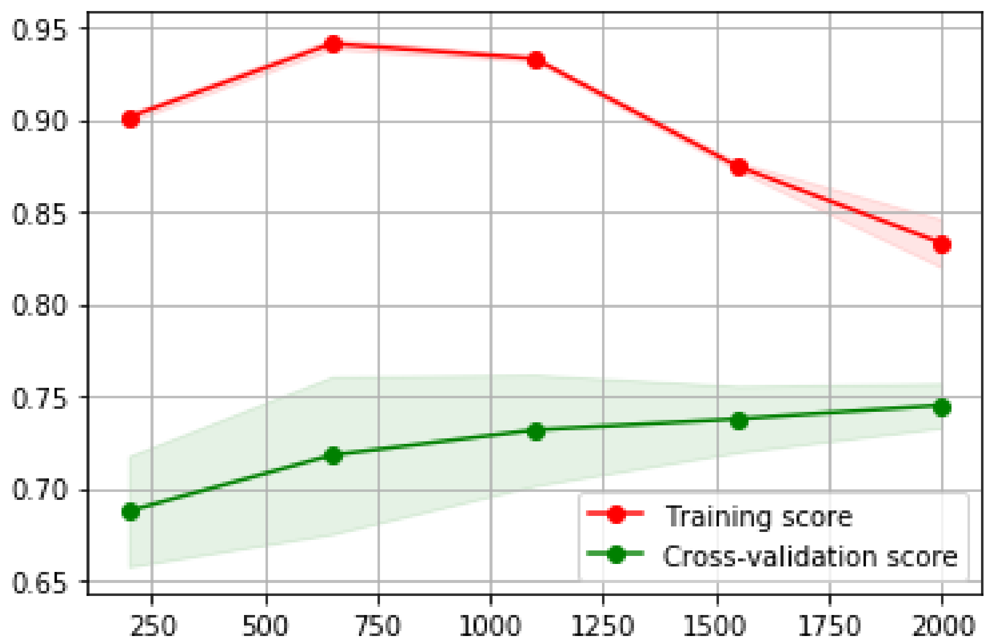

```
'fold 3/4'


accuracy 0.7474949899799599
recall 0.9840213049267643
precision 0.7548518896833504

confusion matrix:
 [[  7 240]
 [ 12 739]]
              precision    recall  f1-score   support

           0       0.37      0.03      0.05       247
           1       0.75      0.98      0.85       751

    accuracy                           0.75       998
   macro avg       0.56      0.51      0.45       998
weighted avg       0.66      0.75      0.66       998
```



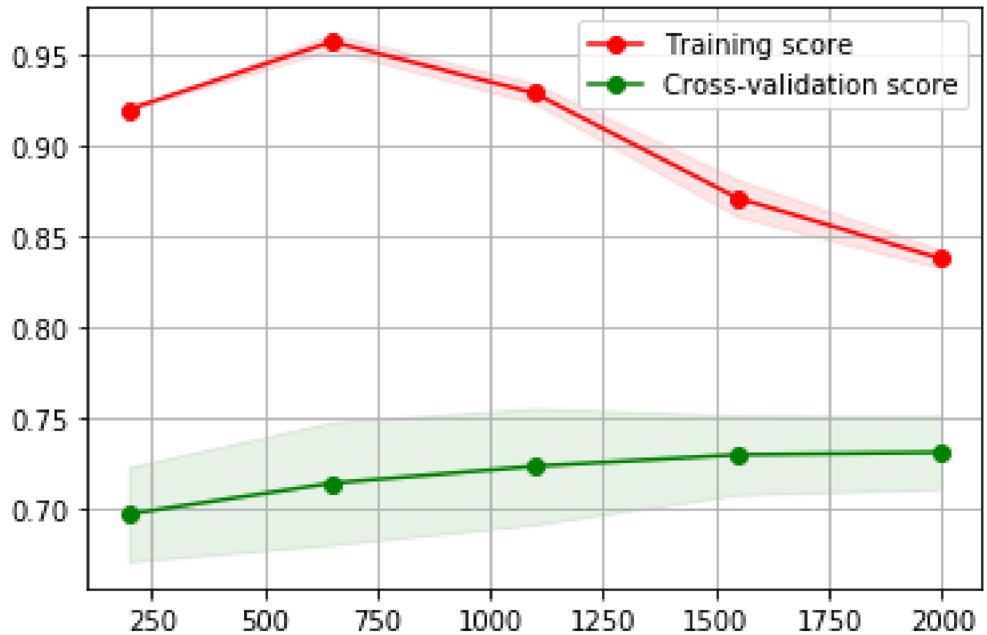

```
'fold 4/4'


accuracy 0.7535070140280561
recall 0.9933422103861518
precision 0.7558257345491388

confusion matrix:
 [[  6 241]
 [  5 746]]
              precision    recall  f1-score   support

           0       0.55      0.02      0.05       247
           1       0.76      0.99      0.86       751

    accuracy                           0.75       998
   macro avg       0.65      0.51      0.45       998
weighted avg       0.70      0.75      0.66       998
```



[21]:
```
# Overview feateure_importances

# Note: the only use in this cell is to identify important features
lgbm_model = LGBMClassifier(
    metric='recall',
    objective='binary',
    n_estimators=50,
    learning_rate=0.05,
    scale_pos_weight=1,
    n_jobs=4,
)
lgbm_model.fit( train_df, enc_train_labels )

# Identify important features
model0_feat_imp = pd.DataFrame({
    'imp_name': train_df.columns,
    'imp_val': lgbm_model.booster_.feature_importance(importance_type='gain')
}).sort_values(by='imp_val', ascending=False)

display(model0_feat_imp.head(10))
display(model0_feat_imp.tail(20))

features_to_drop = model0_feat_imp[model0_feat_imp['imp_val'] < 50]['imp_name'].
 ↪values

display(features_to_drop)
```

```
                  imp_name     imp_val
15        total_time_spent  789.451061
27      days_from_passbdate  733.139058
28  days_from_jobsworksince  713.314591
26          days_from_birth  680.419847
22        weird_sqrcost_inc  642.406919
23       intrct_income_cost  590.620780
16        avg_time_per_page  367.612961
21       weird_sqrtcost_inc  344.531940
14         total_visits_cnt  314.277739
60        binned_empl_type_2  311.889527
```

```
                       imp_name   imp_val
46          visit_top_dayhour_22  22.31243
65       binned_education_area_4  20.27568
33                top_platform_2  20.01955
45          visit_top_dayhour_15  17.14636
59      binned_work_experience_6  16.09795
38         visit_top_dayofweek_5  15.03115
51       binned_quantity_child_1  14.79858
70  binned_days_from_passbdate_3  13.68266
72               app_month_num_9  13.64507
40           visit_top_dayhour_3  13.01157
41           visit_top_dayhour_9  12.91780
54               binned_region_2  12.83861
56           binned_region_reg_2  11.06924
32                 top_browser_7   9.35391
52        binned_max_age_child_2   8.85947
29                   flg_has_job   6.92683
68      binned_days_from_birth_3   6.17060
69      binned_days_from_birth_4   5.84703
39         visit_top_dayofweek_6   4.10441
36         visit_top_dayofweek_1   3.53612
```

```
array(['top_browser_3', 'top_platform_5', 'binned_region_reg_3',
       'app_month_num_10', 'binned_days_from_birth_2',
       'visit_top_dayhour_12', 'binned_empl_worker_count_3',
       'quantity_child', 'binned_work_experience_4', 'visit_days_cnt',
       'binned_education_area_3', 'visit_top_dayhour_22',
       'binned_education_area_4', 'top_platform_2',
       'visit_top_dayhour_15', 'binned_work_experience_6',
       'visit_top_dayofweek_5', 'binned_quantity_child_1',
       'binned_days_from_passbdate_3', 'app_month_num_9',
       'visit_top_dayhour_3', 'visit_top_dayhour_9', 'binned_region_2',
       'binned_region_reg_2', 'top_browser_7', 'binned_max_age_child_2',
       'flg_has_job', 'binned_days_from_birth_3',
```

```
       'binned_days_from_birth_4', 'visit_top_dayofweek_6',
       'visit_top_dayofweek_1'], dtype=object)
```

[22]:
```
# Drop redundant columns and try the same classifier (same params) again

# Results: same as bad

train_df = train_df.drop( features_to_drop, axis=1 )

try_model(
    train_df, enc_train_labels
)
```

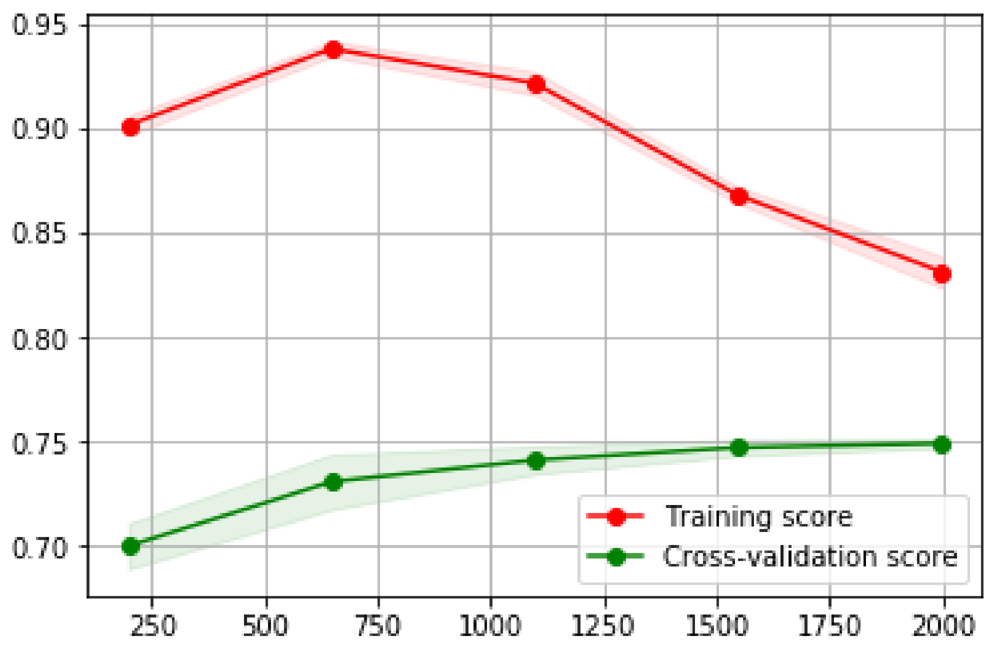
```
'fold 1/4'


accuracy 0.752
recall 0.9986702127659575
precision 0.7525050100200401

confusion matrix:
 [[  1 247]
 [  1 751]]
              precision    recall  f1-score   support

           0       0.50      0.00      0.01       248
           1       0.75      1.00      0.86       752

    accuracy                           0.75      1000
   macro avg       0.63      0.50      0.43      1000
weighted avg       0.69      0.75      0.65      1000
```



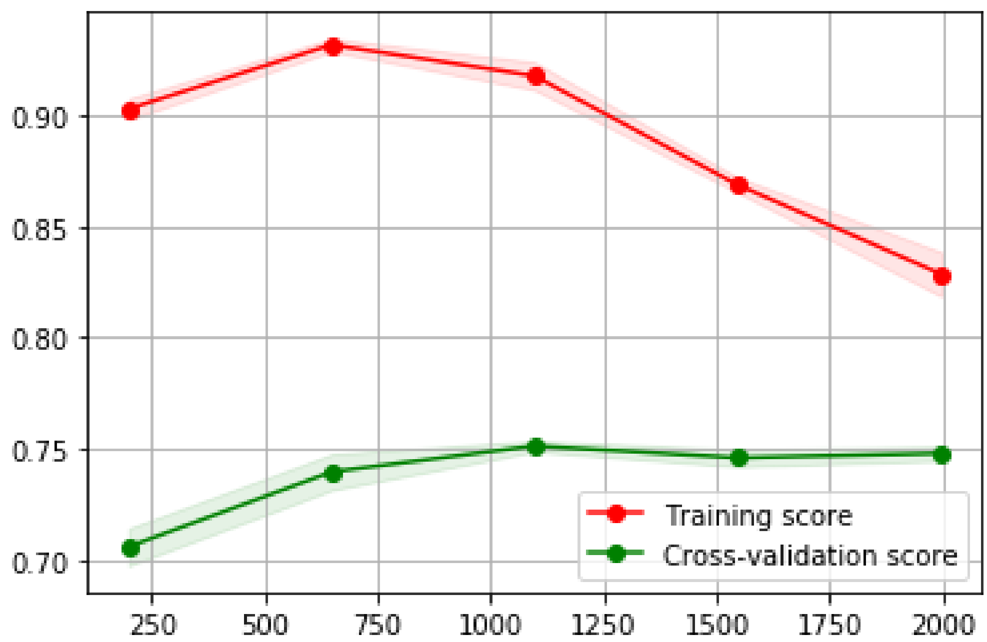

```
'fold 2/4'


accuracy 0.753
recall 0.9960106382978723
precision 0.7542799597180262

confusion matrix:
 [[  4 244]
 [  3 749]]
              precision    recall  f1-score   support

           0       0.57      0.02      0.03       248
           1       0.75      1.00      0.86       752

    accuracy                           0.75      1000
   macro avg       0.66      0.51      0.44      1000
weighted avg       0.71      0.75      0.65      1000
```



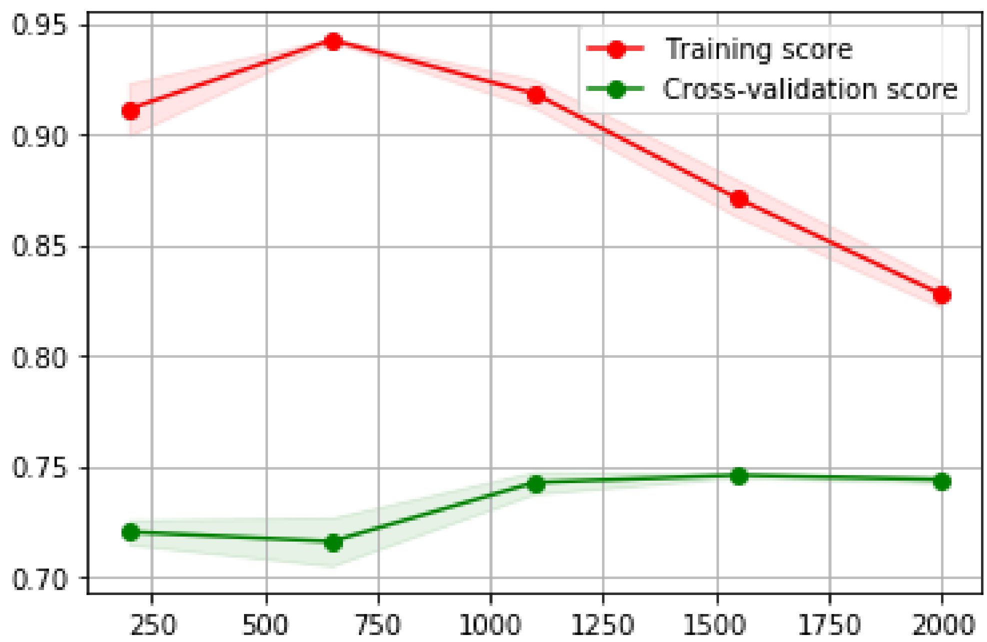

```
'fold 3/4'


accuracy 0.7545090180360722
recall 0.9946737683089214
precision 0.7560728744939271

confusion matrix:
 [[  6 241]
 [  4 747]]
              precision    recall  f1-score   support

           0       0.60      0.02      0.05       247
           1       0.76      0.99      0.86       751

    accuracy                           0.75       998
   macro avg       0.68      0.51      0.45       998
weighted avg       0.72      0.75      0.66       998
```



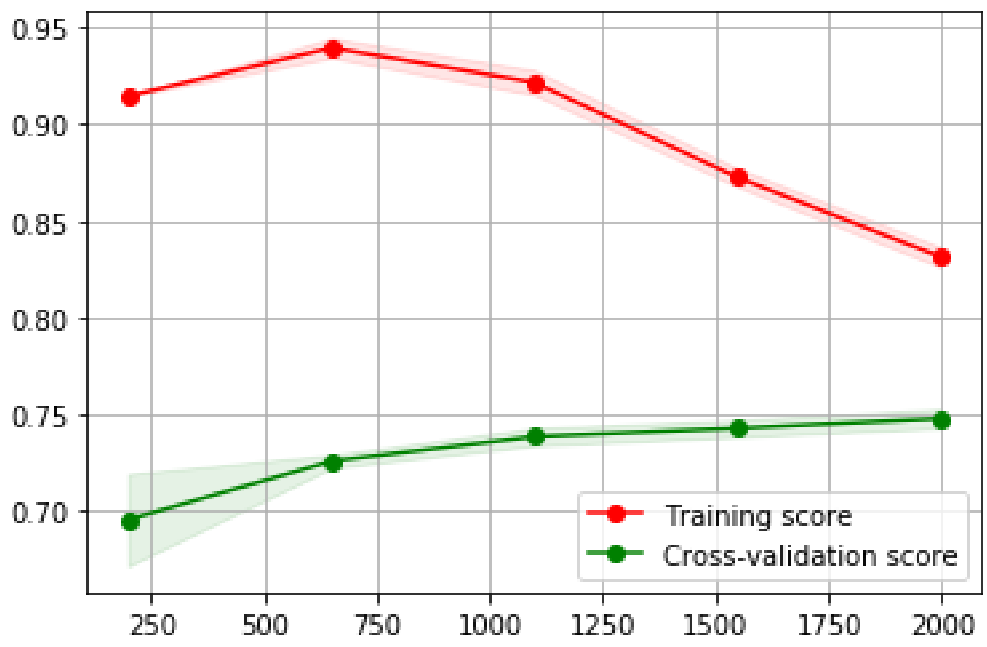

```
'fold 4/4'


accuracy 0.750501002004008
recall 0.9920106524633822
precision 0.7540485829959515

confusion matrix:
 [[  4 243]
 [  6 745]]
              precision    recall  f1-score   support

           0       0.40      0.02      0.03       247
           1       0.75      0.99      0.86       751

    accuracy                           0.75       998
   macro avg       0.58      0.50      0.44       998
weighted avg       0.67      0.75      0.65       998
```



[76]:
```
# Try oversampling

# Use imblearn.over_sampling.SMOTE to resample target feature

def try_model_smote( data_df, target_df ):
    skf = StratifiedKFold(
        n_splits=4,
        shuffle=True
    )

    idx = 0
    for train_idx, val_idx in skf.split(data_df, target_df):
        X_tr, X_val = data_df.iloc[train_idx, :], data_df.iloc[val_idx, :]
        y_tr, y_val = target_df.iloc[train_idx], target_df.iloc[val_idx]

        sm = SMOTE(ratio=0.85)
        X_tr_res, y_tr_res = sm.fit_sample( X_tr, y_tr )

        idx += 1
        display('fold {0}/{1}'.format( idx, skf.get_n_splits() ))

        lgbm_model = LGBMClassifier(
            metric='recall',
            objective='binary',
            max_depth=250,
            learning_rate=0.5,
```

```
            n_estimators=10,
            n_jobs=4,
        )
        lgbm_model.fit(X_tr_res, y_tr_res)

        val_y_pred = lgbm_model.predict(X_val)
        print( 'accuracy', lgbm_model.score(X_val, y_val) )
        print( 'recall', recall_score(y_val, val_y_pred ) )
        print( 'precision', precision_score(y_val, val_y_pred) )
        print( '\nconfusion matrix:\n', confusion_matrix(y_val, val_y_pred) )
        print( classification_report(y_val, val_y_pred) )

        draw_learning_curve( lgbm_model, X_tr_res, y_tr_res )
```

[77]:
```
try_model_smote( train_df, enc_train_labels )
```

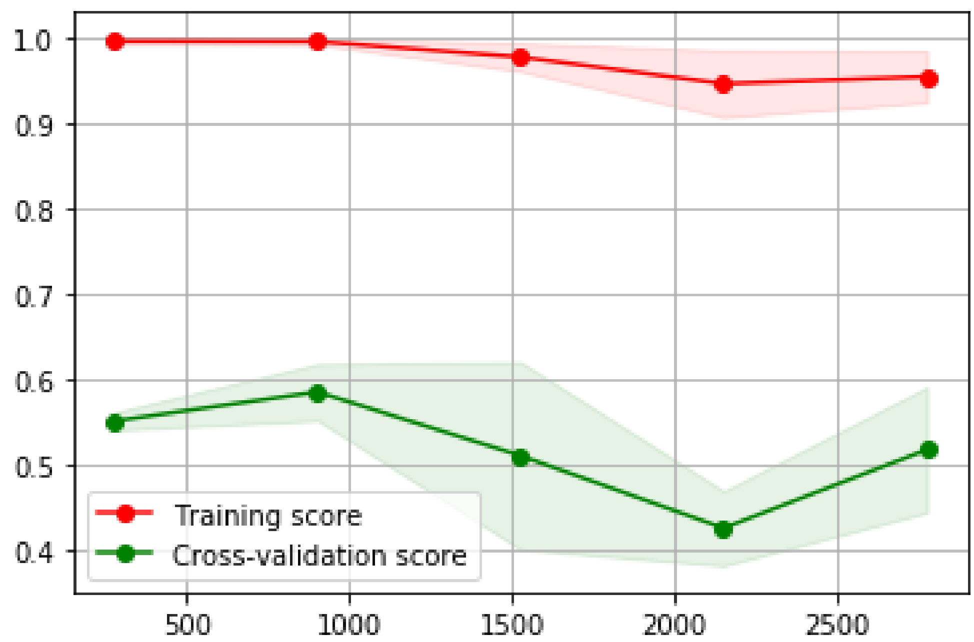
```
'fold 1/4'


accuracy 0.716
recall 0.8949468085106383
precision 0.7665148063781321

confusion matrix:
 [[ 43 205]
 [ 79 673]]
              precision    recall  f1-score   support

           0       0.35      0.17      0.23       248
           1       0.77      0.89      0.83       752

    accuracy                           0.72      1000
   macro avg       0.56      0.53      0.53      1000
weighted avg       0.66      0.72      0.68      1000
```



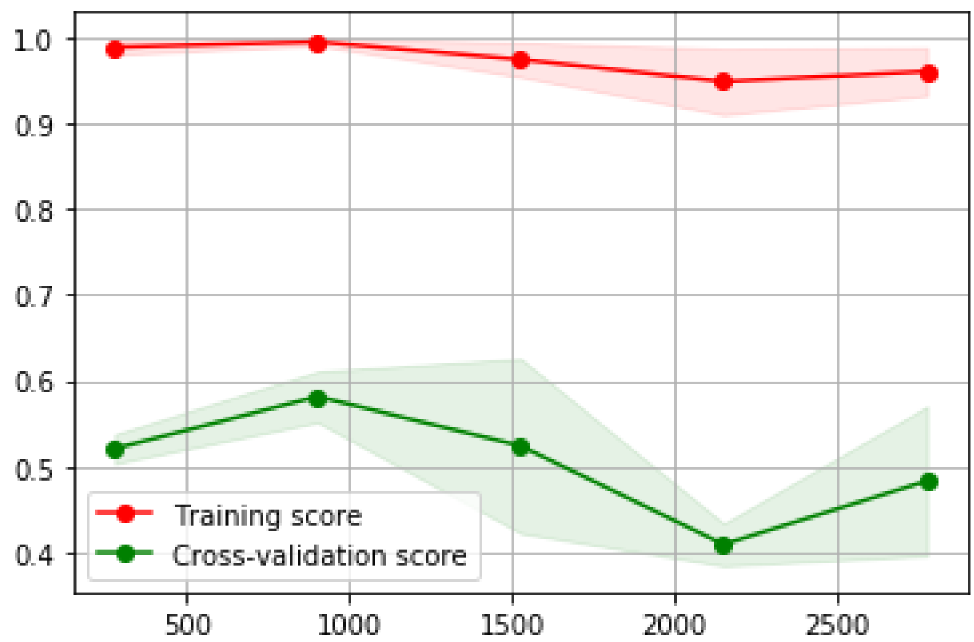

```
'fold 2/4'


accuracy 0.711
recall 0.910904255319149
precision 0.7552370452039692

confusion matrix:
 [[ 26 222]
 [ 67 685]]
              precision    recall  f1-score   support

           0       0.28      0.10      0.15       248
           1       0.76      0.91      0.83       752

    accuracy                           0.71      1000
   macro avg       0.52      0.51      0.49      1000
weighted avg       0.64      0.71      0.66      1000
```



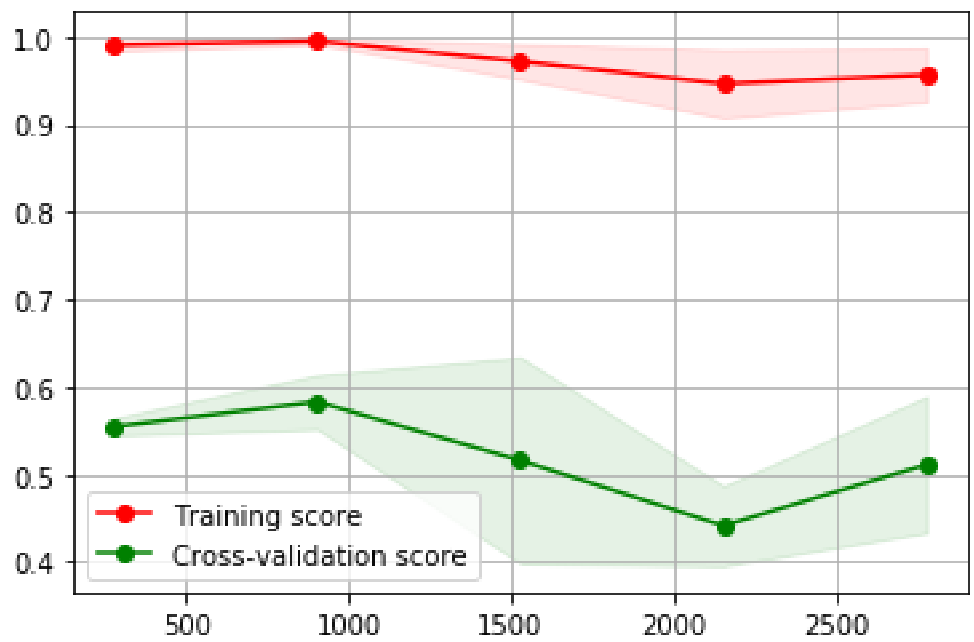

```
'fold 3/4'


accuracy 0.7004008016032064
recall 0.8908122503328895
precision 0.755079006772009

confusion matrix:
 [[ 30 217]
 [ 82 669]]
              precision    recall  f1-score   support

           0       0.27      0.12      0.17       247
           1       0.76      0.89      0.82       751

    accuracy                           0.70       998
   macro avg       0.51      0.51      0.49       998
weighted avg       0.63      0.70      0.66       998
```



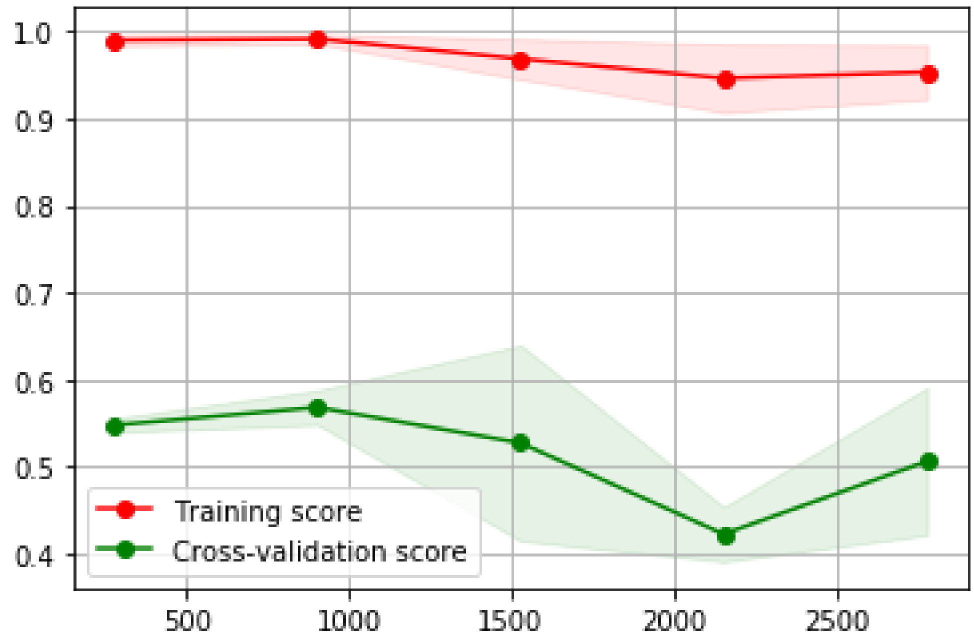

```
'fold 4/4'


accuracy 0.7074148296593187
recall 0.8788282290279628
precision 0.7665505226480837

confusion matrix:
 [[ 46 201]
 [ 91 660]]
              precision    recall  f1-score   support

           0       0.34      0.19      0.24       247
           1       0.77      0.88      0.82       751

    accuracy                           0.71       998
   macro avg       0.55      0.53      0.53       998
weighted avg       0.66      0.71      0.68       998
```


1.0
0.9
0.8
0.7
0.6
0.5
0.4
500
1000
1500
2000
2500
Training score
Cross-validation score