Финальный проект
в рамках специализации Coursera
«Машинное обучение и анализ данных»
by Moscow Institute of Physics and Technology
& Yandex

Прогнозирование оттока клиентов (churn prediction)

В проекте решалась одна из наиболее актуальных задач из области customer relationship managmenet (CRM): прогнозирование оттока пользователей.

Суть задачи заключается в заблаговременном нахождении сегмента пользователей, склонных через некоторый промежуток времени отказаться от услуг оператора сотовой связи.

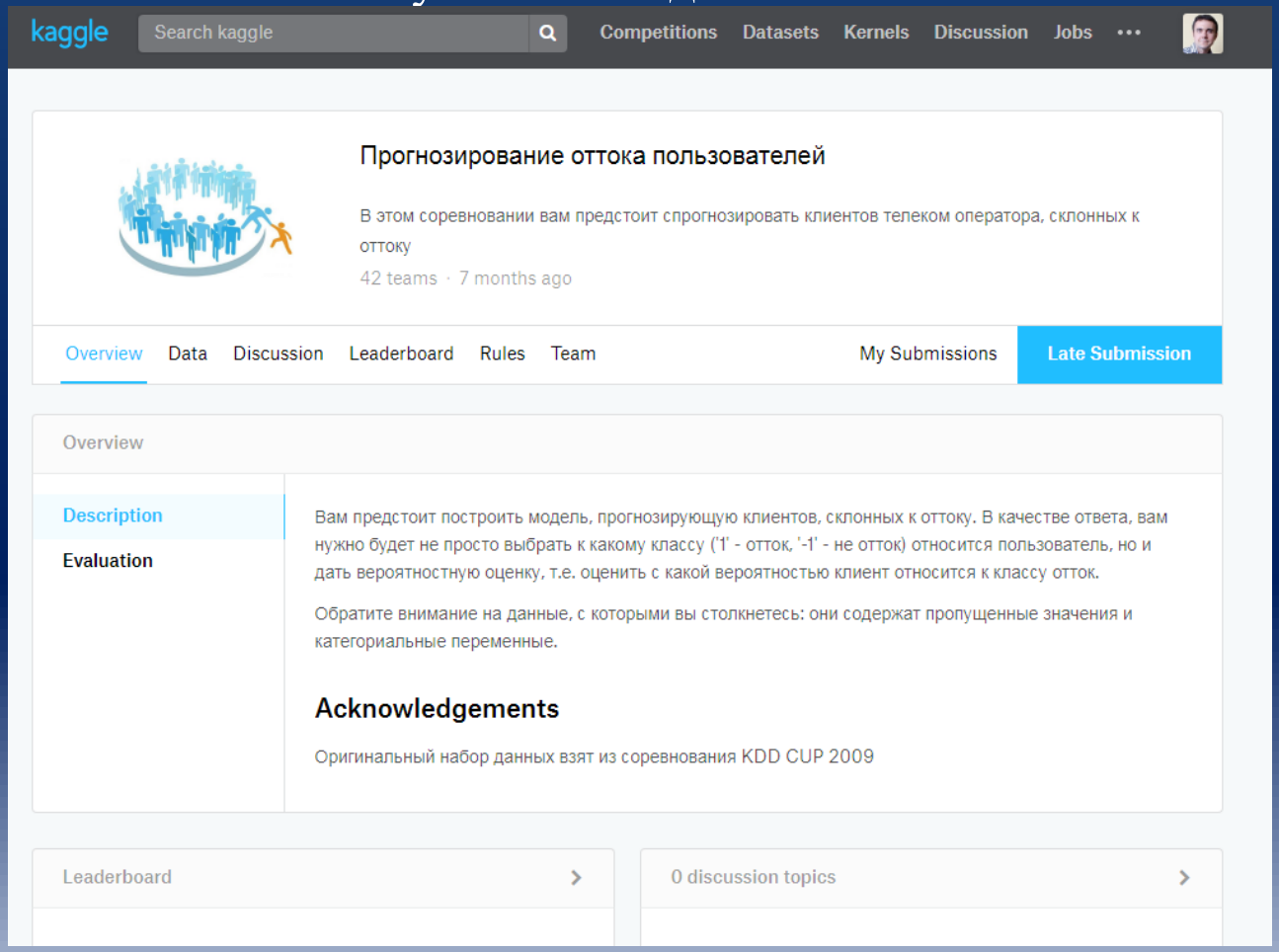
В процессе работы над проектом построена и оптимизирована прогнозная модель, оценено её качество и экономический потенциал.

Критерием качества построенного решения было преодоление базового порога конкурса на сайте Kaggle.com

Базовый порог: 0.69

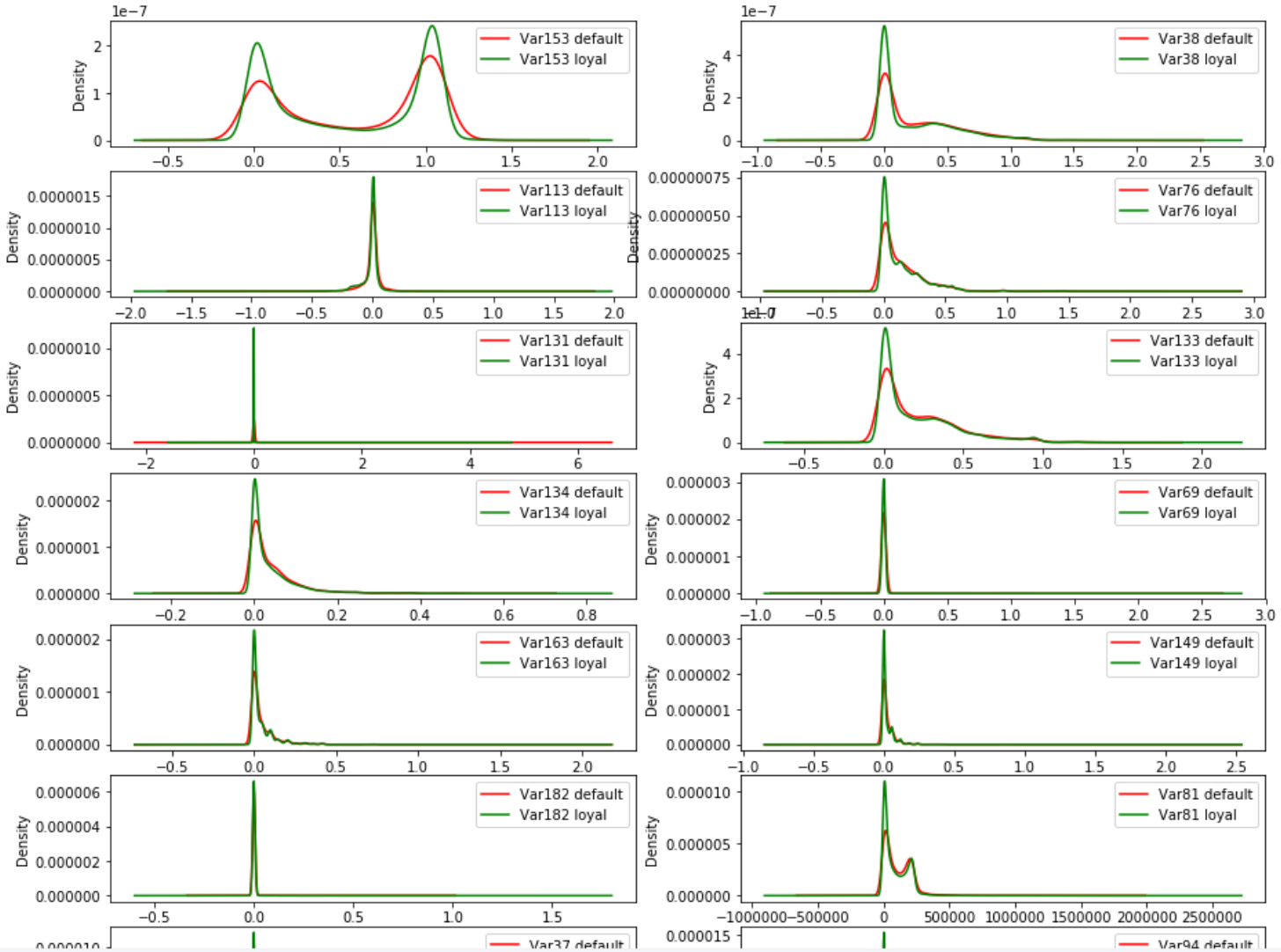
Результат финального решения: 0.74 (Top10)

Результат победителя: 0.81



Отбор признаков, наиболее коррелирующих с целевой переменной

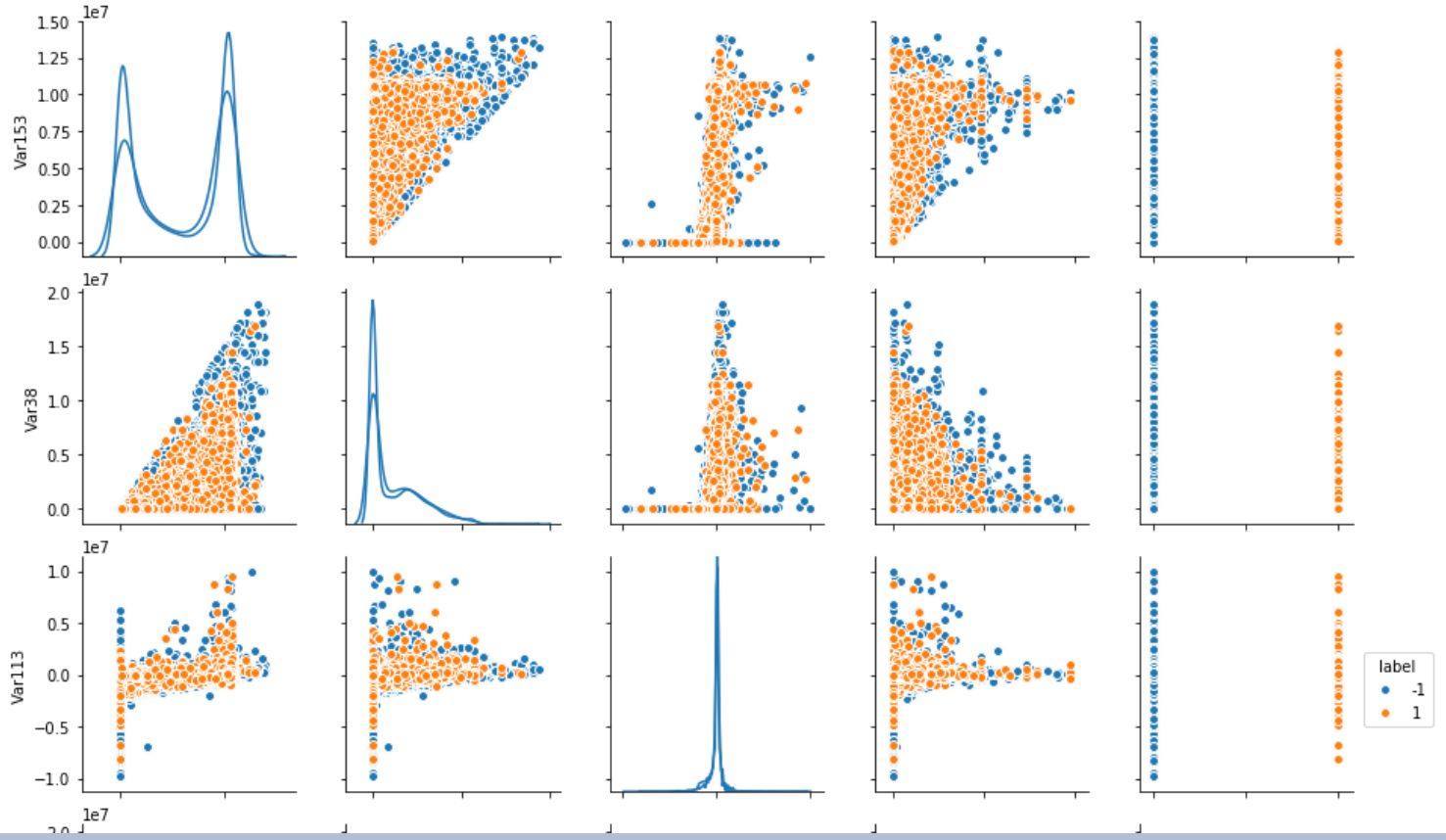
In [24]:
```
fig, axes = plt.subplots(nrows=10, ncols=2, figsize=(15, 20)) # default - отток. loyal - обратное
for col, i in zip (num_20, range(20)):
    pos = X_pos[col]
    neg = X_neg[col]
    pos.plot(ax=axes[i/2, i%2],color='r', subplots=True, label=col + ' default', kind='kde', legend=True)
    neg.plot(ax=axes[i/2, i%2],color='g', subplots=True, label=col + ' loyal', kind='kde', legend=True)
```



Отображение объектов в координатах пар признаков

In [15]:
```
sns_plot = sns.pairplot(Top_4, hue='label',diag_kind='kde', size=2.5)
sns_plot.savefig('Top_4.png')
```

```
C:\ProgramData\Anaconda2\lib\site-packages\statsmodels\nonparametric\kde.py:494: RuntimeWarning: invalid value encountered in d
ivide
  binned = fast_linbin(X,a,b,gridsize)/(delta*nobs)
C:\ProgramData\Anaconda2\lib\site-packages\statsmodels\nonparametric\kde.py:494: RuntimeWarning: invalid value encountered in t
rue_divide
  binned = fast_linbin(X,a,b,gridsize)/(delta*nobs)
C:\ProgramData\Anaconda2\lib\site-packages\statsmodels\nonparametric\kdetools.py:34: RuntimeWarning: invalid value encountered
 in double_scalars
  FAC1 = 2*(np.pi*bw/RANGE)**2
```



Проверка качества предварительной (baseline) модели

In [97]:
```
model_test(linear_model.LogisticRegression(random_state = 0), train, labels.values)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=0, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
Scores and mean for metric accuracy:
[ 0.92411765  0.925       0.92411765  0.92470588  0.92441176  0.92441176
  0.92411765  0.92441176  0.92676471  0.92647059]
0.924852941176
Scores and mean for metric roc_auc:
[ 0.65544323  0.66518712  0.66186254  0.67217791  0.65847893  0.6590077
  0.68455559  0.66336721  0.69426808  0.67371648]
0.668806479852
Scores and mean for metric recall:
[ 0.          0.00395257  0.00790514  0.01581028  0.01185771  0.00395257
  0.01185771  0.          0.01976285  0.01581028]
0.00909090909091
Scores and mean for metric f1_weighted:
[ 0.88908537  0.89009441  0.89019909  0.89158999  0.89089665  0.8897959
  0.89074303  0.88923241  0.89322638  0.89252458]
0.890738780531
------
```

In [106]:
```
model_test(ensemble.RandomForestClassifier(random_state = 0), train, labels.values)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, 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=0, verbose=0, warm_start=False)
Scores and mean for metric accuracy:
[ 0.92470588  0.92470588  0.92529412  0.92529412  0.92529412  0.92529412
  0.92529412  0.925       0.92558824  0.92441176]
0.925088235294
Scores and mean for metric roc_auc:
[ 0.59987553  0.60603925  0.56670887  0.5828501   0.56084093  0.54764309
  0.59737362  0.56089305  0.61081637  0.61806526]
0.585110607882
Scores and mean for metric recall:
[ 0.00395257  0.00395257  0.          0.          0.00395257  0.00790514
  0.00790514  0.          0.00395257  0.00395257]
0.00355731225296
Scores and mean for metric f1_weighted:
[ 0.88994517  0.88994517  0.88967326  0.88967326  0.89024363  0.89080497
  0.89080497  0.88952636  0.89039282  0.8897959 ]
```

Использование Pipeline для обработки разных типов признаков

In [18]:
```
model = xgb.XGBClassifier(max_depth=7, learning_rate=0.05, n_estimators=250, silent=True, objective='binary:logistic',
                          nthread=-1, gamma=0, min_child_weight=2,subsample=0.6, colsample_bytree = 0.6, reg_alpha=0, reg_lambda=1,
                          scale_pos_weight=0.04, base_score=0.5, seed=0, missing=None)
```

In [19]:
```
nums_ind = np.array([(column in nums) for column in train.columns], dtype = bool)
cats_ind = np.array([(column in cats) for column in train.columns], dtype = bool)
```

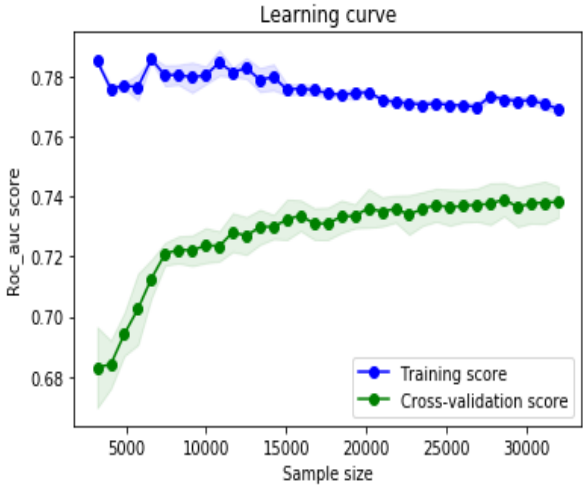
In [22]:
```
estimator = pipeline.Pipeline(steps = [
    ('processing', pipeline.FeatureUnion(transformer_list = [
        #nums
        ('nums_processing', pipeline.Pipeline(steps = [
                ('selecting', pp.FunctionTransformer(lambda x: x[:, nums_ind])),
                ('scaling', pp.StandardScaler(with_mean=0))
                        ])),
        #cats
        ('cats_processing', pipeline.Pipeline(steps = [
            ('selecting', pp.FunctionTransformer(lambda x: x[:, cats_ind])),
            ('encoding', pp.OneHotEncoder(handle_unknown = 'ignore'))
        ]))

    ])),
    ('model_fitting', model)
])
```

In [23]:
```
estimator.fit(train, y)
```

Out[23]:
```
Pipeline(memory=None,
     steps=[('processing', FeatureUnion(n_jobs=1,
       transformer_list=[('nums_processing', Pipeline(memory=None,
     steps=[('selecting', FunctionTransformer(accept_sparse=False,
          func=<function <lambda> at 0x7f52066ce2f0>, inv_kw_args=None,
          inverse_func=None, kw_args=None, pass_y...tic', reg_alpha=0, reg_lambda=1,
       scale_pos_weight=0.04, seed=0, silent=True, subsample=0.6))])
```

Оценка размера тестовой выборки для улучшения качества модели

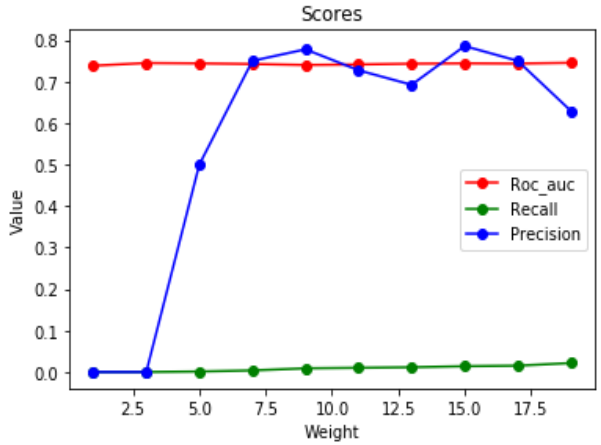
In [9]:
```
plt.figure()
plt.title("Learning curve")
plt.xlabel("Sample size")
plt.ylabel("Roc_auc score")
train_sizes, train_scores, test_scores = learning_curve(
        est, X_train, y, cv=5, train_sizes=np.linspace(0.1, 1, 35),
        scoring='roc_auc')
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)
plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                    train_scores_mean + train_scores_std, alpha=0.1,
                    color="b")
plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                    test_scores_mean + test_scores_std, alpha=0.1, color="g")
plt.plot(train_sizes, train_scores_mean, 'o-', color="b",
             label="Training score")
plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation score")
plt.legend(loc="best")
plt.show()
```



Настройка весов (обучающая выборка сильно разбалансирована)

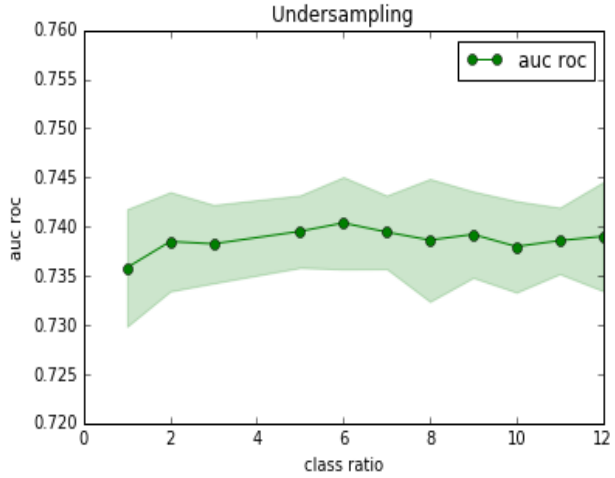
In [10]:
```
train_x, test_x, train_y, test_y = train_test_split(X_train, y, random_state = 42)
weights_sequence = range(1, 21, 2)
precision, recall, roc_auc = [], [], []
for weight in weights_sequence:
    weights = [1 if train_y.values[x] == -1 else weight for x in range(train_x.shape[0])]
    est.fit(train_x, train_y, sample_weight = weights)
    values = est.predict_proba(test_x)[:,1]
    pred = est.predict(test_x)
    roc_auc.append(metrics.roc_auc_score(test_y, values))
    recall.append(metrics.recall_score(test_y, pred))
    precision.append(metrics.precision_score(test_y, pred))

plt.figure()
plt.title("Scores")
plt.xlabel("Weight")
plt.ylabel("Value")
plt.plot(weights_sequence, roc_auc, 'o-', color="r",
         label="Roc_auc")
plt.plot(weights_sequence, recall, 'o-', color="g",
         label="Recall")
plt.plot(weights_sequence, precision, 'o-', color="b",
         label="Precision")
plt.legend(loc="best")
plt.show()
```

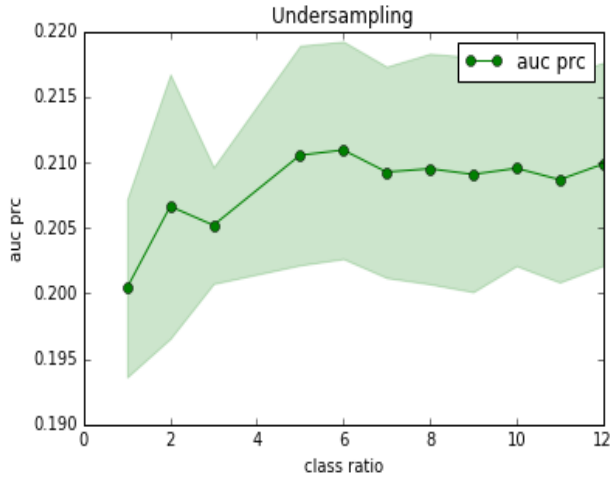
```
/resources/common/.virtualenv/python2/lib/python2.7/site-packages/sklearn/metrics/classification.py:1135: UndefinedMetricWarnin
g: Precision is ill-defined and being set to 0.0 due to no predicted samples.
  'precision', 'predicted', average, warn_for)
```



Применение undersampling

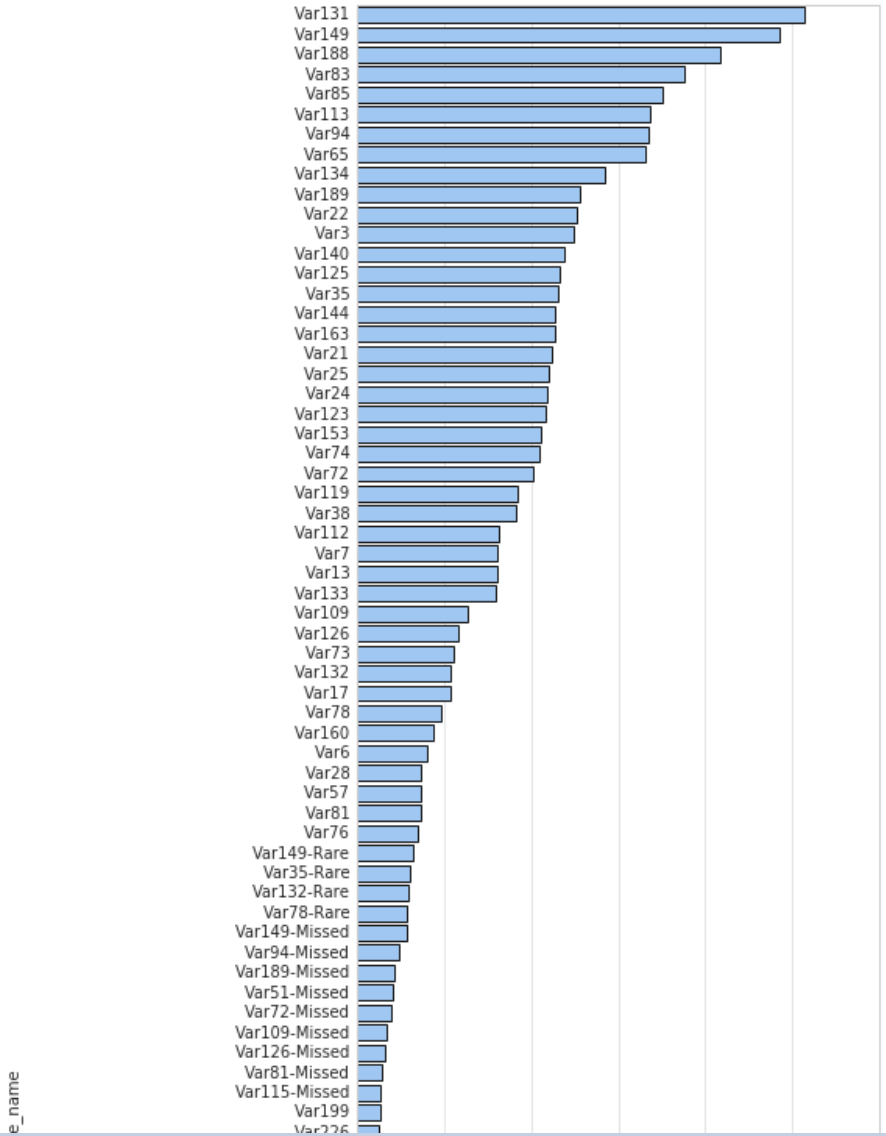
In [18]:
```
plot_curve(undersam_params, auc_roc_scores,
           'Undersampling',
           'class ratio', 'auc roc', ylim=(0.72,0.76))
plot_curve(undersam_params, auc_prc_scores,
           'Undersampling',
           'class ratio', 'auc prc')
```





Отбор признаков, внесших ненулевой вклад в модель

In [67]:
```
important_feat = X.columns[clf.feature_importances_ > 0.0].tolist()
pos_import = clf.feature_importances_[clf.feature_importances_ > 0.0]
sorted_index = np.argsort(pos_import)[::-1]
plot_importance(pos_import[sorted_index], important_feat[::-1])
```



Построение финальной модели и подготовка результата для конкурса

In [29]:
```
def make_submission(model, X_test, file_name):
    submission = pd.DataFrame({'result': np.zeros((10000))})
    submission.result = model.predict_proba(X_test)[:,1]
    submission.to_csv(file_name + '.csv', index_label='Id')
```

In [30]:
```
clf = xgb.XGBClassifier(max_depth=7, learning_rate=0.05, n_estimators=250, silent=True, objective='binary:logistic',
                        nthread=-1, gamma=0, min_child_weight=2,subsample=0.6, colsample_bytree = 0.6, reg_alpha=0,
                        reg_lambda=0.75,scale_pos_weight=0.08, base_score=0.5, seed=0, missing=None)
```

In [32]:
```
clf.fit(X_train,y)
```

Out[32]:
```
XGBClassifier(base_score=0.5, colsample_bylevel=1, colsample_bytree=0.6,
       gamma=0, learning_rate=0.05, max_delta_step=0, max_depth=7,
       min_child_weight=2, missing=None, n_estimators=250, nthread=-1,
       objective='binary:logistic', reg_alpha=0, reg_lambda=0.75,
       scale_pos_weight=0.08, seed=0, silent=True, subsample=0.6)
```

In [33]:
```
important_feat = X.columns[clf.feature_importances_ > 0.0].tolist()
```

In [34]:
```
X_impr = X_train[important_feat]
```

In [35]:
```
clf.fit(X_impr,y)
```

Out[35]:
```
XGBClassifier(base_score=0.5, colsample_bylevel=1, colsample_bytree=0.6,
       gamma=0, learning_rate=0.05, max_delta_step=0, max_depth=7,
       min_child_weight=2, missing=None, n_estimators=250, nthread=-1,
       objective='binary:logistic', reg_alpha=0, reg_lambda=0.75,
       scale_pos_weight=0.08, seed=0, silent=True, subsample=0.6)
```

In [36]:
```
X_test_impr = X_test[important_feat]
```

In [37]:
```
make_submission(clf, X_test_impr, 'final')
```

Прогнозирование оттока пользователей

В этом соревновании вам предстоит спрогнозировать клиентов телеком оператора, склонных к оттоку

42 teams · 7 months ago

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
final.csv	2 minutes ago	3 seconds	0 seconds	0.70551

Complete

Jump to your position on the leaderboard

You can select up to 2 submissions to be used to calculate your final leaderboard score. If 2 submissions are not selected, they will be chosen based on your best submission scores on the public leaderboard.

Your final score may not be based on the same exact subset of data as the public leaderboard, but rather a different private data subset of your full submission — your public score is only a rough indication of what your final score is.

You should thus choose submissions that will most likely be best overall, and not necessarily on the public subset.

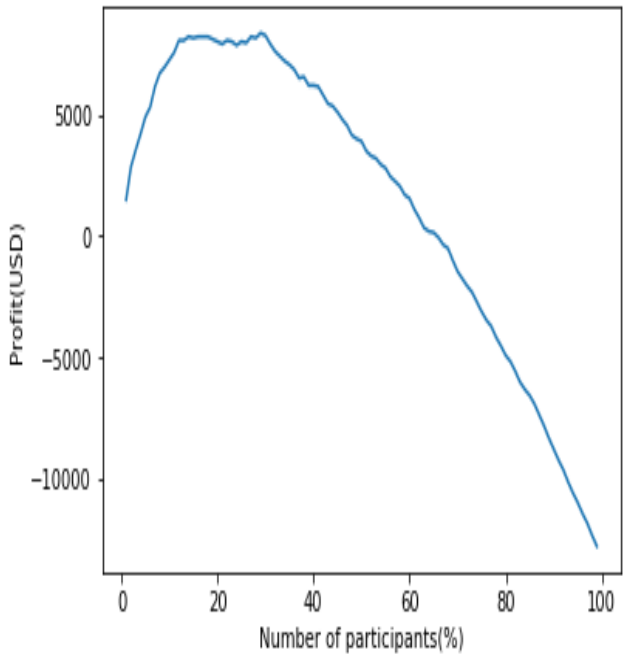
59 submissions for Vladimir Gmyzin Sort by Most recent

All Successful Selected

Submission and Description	Private Score	Public Score	Use for Final Score
final.csv 2 minutes ago by Vladimir Gmyzin add submission details	0.74080	0.70551	☐

Оценка потенциального экономического эффекта от внедрения полученного решения

In [9]:
```
rev = []
for per_cent in range(1, 100, 1):
    rev.append(econ_model(clf, X_test, y_test, 100*0.05, per_cent, 0.50, 100))
plot(range(1, 100, 1), rev)
res = sorted(zip(rev, range(1, 100, 1)))
print("Best participants number and profit : %d percent - %d USD for 10 000 patricipants." % (res[-1][1], res[-1][0]))
```




```
Best participants number and profit : 29 percent - 8350 USD for 10 000 patricipants.
```