Анализ данных и машинное обучение

Финальный проект

# Прогнозирование оттока пользователей

Автор:

Александр Сааков

26 апреля 2021 г.

# Содержание



# 1 Описание проекта

В 2009 году проводилось соревнование KDD Cup: Customer relationship prediction. В рамках этого соревнования участникам предлагалось решить 3 задачи из области Customer Relationship Management (CRM):
- оценка вероятности того, что клиент осуществит переход к конкуренту (churn prediction);
- оценка склонности клиента к покупке новых продуктов и услуг (appetency);
- оценка склонности клиента к покупке обновлений или дополнений к ранее приобретенным продуктам (up-selling).

Подробно с соревнованием можно ознакомиться на сайте KDD: http://www.kdd.org/kdd-cup/view/kdd-cup-2009/Intro

Данные для соревнования были предоставлены французской телекоммуникационной компанией Orange. В задаче речь идет о клиентских данных, поэтому данные были предварительно обфусцированы и анонимизированны: из датасета убрана любая персональная информация, позволяющая идентифицировать пользователей, а также не представлены названия и описания переменных, предназначенных для построения прогнозов. Мы будем работать с набором данных orange small dataset. Он состоит из 50 тыс. объектов и включает 230 переменных, из которых первые 190 переменных - числовые, и оставшиеся 40 переменные - категориальные.

В рамках проекта мы будем решать задачу прогнозирования оттока пользователей, или churn prediction. Эта задача заключается в прогнозировании вероятности того, что клиент перестанет пользоваться сервисом в течение некоторого заданного заранее промежутка времени, например, решит перейти к конкуренту или откажется от использования услуги данного типа вообще. Задача прогнозирования оттока является одной из важнейших подзадач в области работы с аудиторий и актуальна не только для телекоммуникационных компаний, но и для большинства организаций, оказывающих услуги в сегменте B2C (прим. часто и в

B2B тоже, однако в этом случае под клиентом мы понимаем компанию). Такие задачи часто возникают на практике у телекоммуникационных операторов, провайдеров кабельного телевидения, страховых компаний, банков, крупных и средних интернет-порталов и др.

### 1.1 Цели проекта

Выявить абонентов, склонных перестать пользоваться услугами компании. Если научиться находить таких пользователей с достаточной точностью заблаговременно, то можно эффективно управлять оттоком: например, выявлять причины оттока; помогать пользователям, попавшим в группу риска, решать их проблемы и задачи; проводить кампании по удержанию.
Для того, чтобы находить пользователей, склонных к оттоку, строят прогнозные модели - модели, позволяющие прогнозировать вероятность того, что пользователь покинет сервис. В классической постановке строятся вероятностные модели бинарной классификации, где целевой класс представляют собой пользователи, покидающие сервис. Вероятность того, что пользователь принадлежит целевому классу и есть целевая величина - вероятность оттока. Соответственно, чем эта вероятность больше, тем больше шансов, что пользователь откажется от использования нашего сервиса.
План проекта:
1. Провести анализ данных абонентов сотовой связи
2. Построить модель на основе подготовленных и анализированных данных
3. Выявить абонентов склонных покинуть компанию
4. Построить на основе полученных данных экономическую модель
5. Оценить результаты работы

При работе над проектом мы научимся:
Анализировать анонимизированные данные и использовать их для построения моделей;
Работать с набором разнотипных шумных данных;
Строить и оптимизировать вероятностные модели классификации

## 2 Небольшое предисловие

Естественно, в отчёте не будет подробного описания тех или иных действий. Всех их можно найти в репозитории по ссылке: https://github.com/alexsaakov/Churn-prediction

## 3 Описательный анализ данных

1. Рассчитаны доли классов “отток” (1) и “не отток” (-1).

```
1 labels.label.value_counts()/len(labels)
```

```
-1    0.9256
 1    0.0744
Name: label, dtype: float64
```

2. Рассчитаны корреляции переменных с целевой переменной.

3. Для 20-ти числовых переменных, наиболее сильно коррелирующих с целевой переменной, построены:
а) Распределения в разрезе классов
б) Отображения объектов в координатах пар признаков
4. Для 10-ти случайно выбранных числовых переменных, построены:
а) Распределения в разрезе классов
б) Отображения объектов в координатах пар признаков
5. Для 10-ти числовых переменных, наименее сильно коррелирующих с целевой переменной, построены:
а) Распределения в разрезе классов
б) Отображения объектов в координатах пар признаков
6. Построены гистограммы значений категориальных переменных для разных классов
7. Полученные результаты проанализированы:
Много шумовых данных, неинформативных, а также признаков, коррелирующих между собой. Необходим отбор признаков. Закономерности есть как и в числовых переменных, так и в категориальных. Var177, Var114, Var36 - есть корреляция.
Предположения о наиболее полезных для классификации переменных:
Из числовых переменных: Var131, Var69, Var53, (Var177, Var114, Var36) - есть корреляция этих признаков, Var139
Из категориальных: Var218, Var225
Предположения о наименее полезных, шумовых для классификации переменных:
Переменные в которых есть данные только по label=-1(No churn), коррелирующие переменные, переменные только с одной категорией, например Var191, Var224, Var215, Var213

## 4 Подготовка данных для построения модели

Начнем с простого, но важного шага. Отделим небольшую выборку от существующих данных. Назовем её hold-out dataset. Эта выборка нужна для контроля качества решения: она не должна использоваться вплоть до контроля качества решения. Наличие такой выборки поможет убедиться, что в процессе моделирования не было допущено ошибок, не произошло переобучение.
Три способа обработки категориальных признаков, для того, чтобы их можно было использовать при построении модели.
1) Отображение категориального признака на множество натуральных чисел.
2) Простой OneHot. Каждый категориальный признак превратить в n бинарных
3) OneHot. Аналогично, но теперь n-1 бинарных признаков.

- Метрика качества для наилучшей оценки качества будущей модели: **классическая ROC-AUC.**
- Вспомогательные метрики качества для более полной оценки качества модели: **recall** и **precision**. В принципе, можно посчитать и среднее гармоническое этих двух показателей, то есть **f-меру**.
- Оптимальная стратегия кросс-валидации: **стратифицированный размешанный k-**

**fold с пятью фолдами**. Классы представлены не одинаково. Для 40000 записей оптимально пять фолдов.

## 5 Построение baseline-решений

Теперь нам предстоит построить несколько моделей и оценить их качество. Эти модели будут служить нам в качестве baseline-решений и пригодятся сразу для нескольких задач:

- Во-первых, на разработку baseline-модели не должно уходить много времени (это требование исходит из оценок затрат на проект в целом - большую часть времени все же нужно потратить на основное решение), процесс должен быть простым, на подавляющем большинстве этапов должны использоваться готовые протестированные инструменты. Все это приводит к тому, что baseline-модели - это дешевый способ сделать грубую оценку потенциально возможного качества модели, при построении которого вероятность допущения ошибок относительно невелика.
- Во-вторых, использование моделей разного типа при построении baseline'ов позволяет на раннем этапе сделать предположения о том, какие подходы являются наиболее перспективными и приоритизировать дальнейшие эксперименты.
- Наличие baseline-моделей позволяет оценить, какой прирост качества дают различные преобразования, усложнения, оптимизации и прочие активности, которые вы предпринимаете для построения финального решения.
- Наконец, если после построение сложного решения оценка его качества будет очень сильно отличаться от оценки качества baseline-моделей, то это будет хорошим поводом поискать в решении ошибки.

Обучите 3 разные baseline-модели на полученных наборах данных и оцените их качество. На прошлой неделе вы выбрали методику оценки качества моделей на основе кросс-валидации, а также основную и вспомогательные метрики. Оцените с их помощью получившуюся модель. Напоминаем, что отложенная выборка (hold-out dataset) не должна использоваться для построения и оценки baseline-моделей!
Для начала я попробовал заполнить nanы в данных средним значением:

**Наполнение nan средним значением**

```
1 X_mean = X_full.apply(lambda x: x.fillna(x.mean()),axis=0)
```

```
1 lr = LogisticRegression(random_state=1)
2 res = cross_val_score(lr, X_mean, y, cv=5, scoring='roc_auc')
3 res.mean()
```

```
0.5317316564044091
```

Затем это же проделал с помощью медианных значений:

## Наполнение nan медианой

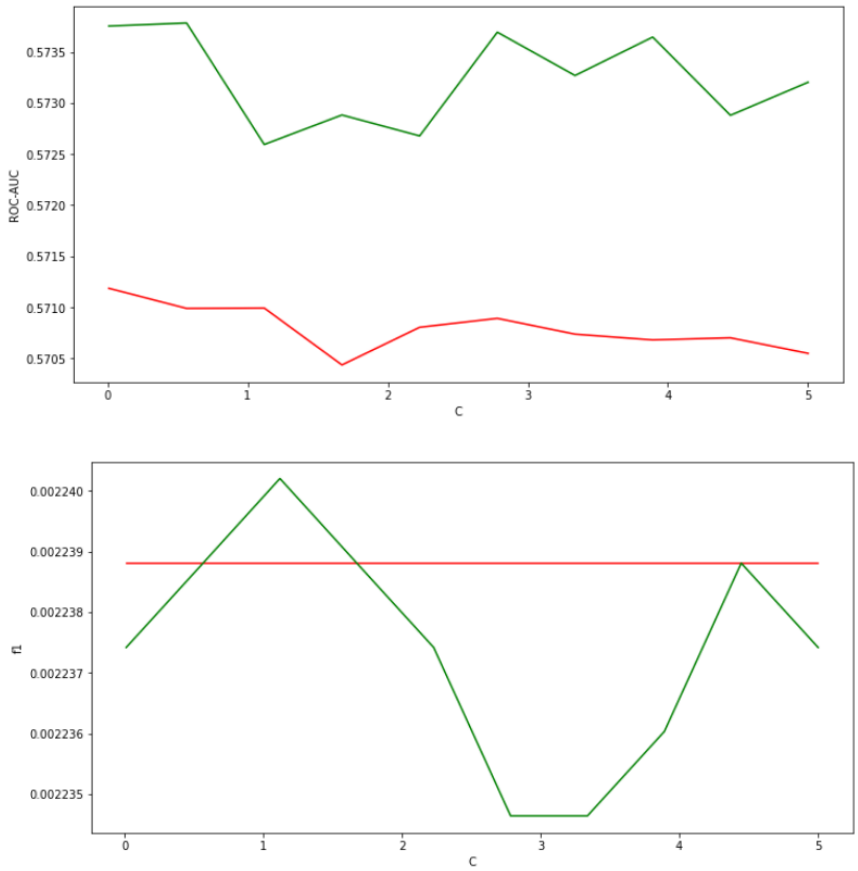
```
X_med = X_full.apply(lambda x: x.fillna(x.median()),axis=0)
```

```
lr = LogisticRegression(random_state=1)
res = cross_val_score(lr, X_med, y, cv=5, scoring='roc_auc')
res.mean()
```

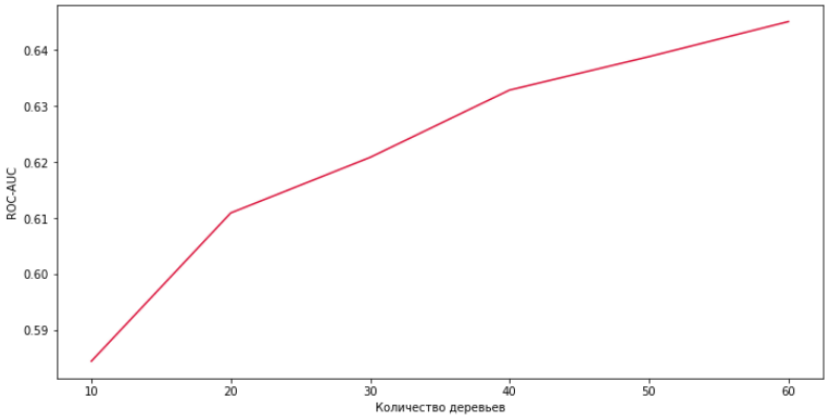
```
0.5280706466825189
```

Ну и после чего решил наполнять nanы средним значением.

Пришло время строить baseline-модели!
Начал я с линейной модели и вот что я получил для метрик roc-auc и f1:
Не сильно огорчившись низким качеством (ведь я от линейной модели высокого и не ожи-




дал) перешел к построению random forest.
Заметил, что эффективность алгоритма потихоньку сходится, поэтому решил остановиться на 100 деревьях:
Для метрик roc-auc и f1 были получены следующие результаты:




```
print('Roc-Auc for RandomForest:', res.mean())
print('F1 for RandomForest:', res_f.mean())
```

```
Roc-Auc for RandomForest: 0.6548954860902102
F1 for RandomForest: 0.0014967272218138542
```

Ну что ж, уже лучше. Самое время попробовать градиентный бустинг, например реализацией XGBClassifier из библиотеки xgboost. Были получены следующие результаты:

```
print('F1 for XGBClassifier:', res_f.mean())
```

```
F1 for XGBClassifier: 0.03247736424617313
```

```
print('Roc-Auc for XGBClassifier:', res.mean())
```

```
Roc-Auc for XGBClassifier: 0.682083104766708
```

Это уже приличный результат! Даже F1-метрика показала себя с лучшей стороны. Замечу, что ещё не было работы с категориальными признаками.

**Резюмируем.** Самую наивысшую оценку мы получили, путём конструирования нашей матрицы через только числовые признаков и выкидывания полностью пустых признаков. Далее неизвестные значения были заменены на медианы. Окончательная матрица обучилась на известном XGBClassifier.

## 6 Построение и оптимизация модели

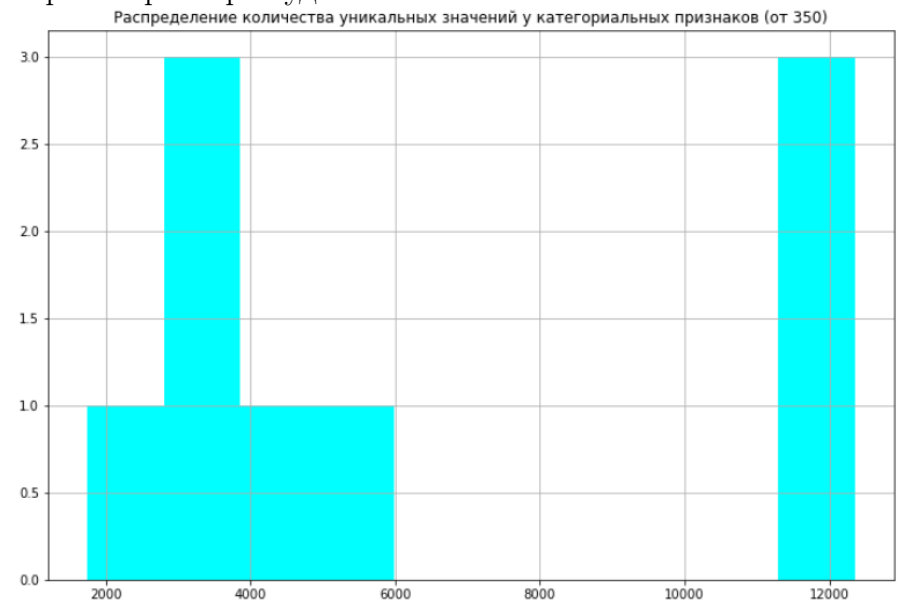
На этом этапе нам предстоит поучаствовать в соревновании на kaggle inclass
Перейдите по ссылке на страницу соревнования:
https://inclass.kaggle.com/c/telecom-clients-prediction2
В соревновании мы будем работать с той же выборкой, что и ранее, поэтому воспользуемся результатами, полученными на предыдущих неделях. Для успешного участия в соревновании необходимо преодолеть по качеству baseline решение.

**Трансформация данных**

Для числовых значений всё просто - мы просто заскейлим их, как обычно. Но что делать с

категориальными? Понятное дело, что мы будем использовать onehotencoding на большинстве из них, однако что делать с данными с большим количеством категорий? Предлагаю любопытный метод:
1. Нумерация разных категорий разными числами
2. Представление чисел в бинарной форме
3. Создание на основе этих бинарных чисел бинарных категорий
Этот метод прекрасен тем, что позволяет нам закодировать n категорий не n - 1 столбцом, а логарифмом по основанию два числа n. Например, для 1024 категорий достаточно 10 столбцов, потому что 1024 - это 2 в десятой.
Для начала нам надо разбить категории на малые и большие, посмотрим на распределение.


Мерилом размера будем использовать число 18.



```
small_columns = categorical_columns[X_categorical.nunique()<=18]
big_columns = categorical_columns[X_categorical.nunique()>18]
```

```
preprocessor = ColumnTransformer(
    transformers=[
        ('nc', StandardScaler(), numeric_columns),
        ('sc', OneHotEncoder(handle_unknown="ignore"), small_columns),
        ('bc', BinaryEncoder(handle_unknown="value"), big_columns)
    ])
```

### XGBoost

```
pipe = Pipeline([('prep', preprocessor),
                 ('xgbc', XGBClassifier(n_estimators=12, n_jobs=-1, objective='binary:logistic',
                                        eval_metric='logloss'))])
pipe.fit(X_train, y_train)
labels = pipe.predict(X_test)
y_test = pipe.predict_proba(X_test)[:, 1]
test['result'] = y_test
assignment = test[['ID', 'result']]
assignment.to_csv('assignment_xgboost.csv', index=False)
```

**Score = 0.70848**

Но на всякий случай попробуем прогнать ещё и логистическую регрессию с random forest. Предполагается, что наша гипотеза о том, что они работают в нашей задаче хуже, чем XGBoost, подтвердится.

### Logistic Regression

```
pipe = Pipeline([('prep', preprocessor), ('lr', LogisticRegression(C=0.2))])
pipe.fit(X_train, y_train)
labels = pipe.predict(X_test)
y_test = pipe.predict_proba(X_test)[:, 1]
test['result'] = y_test
assignment = test[['ID', 'result']]
assignment.to_csv('assignment_lr.csv', index=False)
```

**Score = 0.67773**

### Random Forest

```
pipe = Pipeline([('prep', preprocessor),
                 ('xgbrfc', XGBRFClassifier(n_estimators=120, n_jobs=-1, objective='binary:logistic',
                                            eval_metric='logloss'))])
pipe.fit(X_train, y_train)
labels = pipe.predict(X_test)
y_test = pipe.predict_proba(X_test)[:, 1]
test['result'] = y_test
assignment = test[['ID', 'result']]
assignment.to_csv('assignment_xgbrf.csv', index=False)
```

**Score = 0.69673**

Более тщательный отбор параметров несколько улучшает score в random forest:

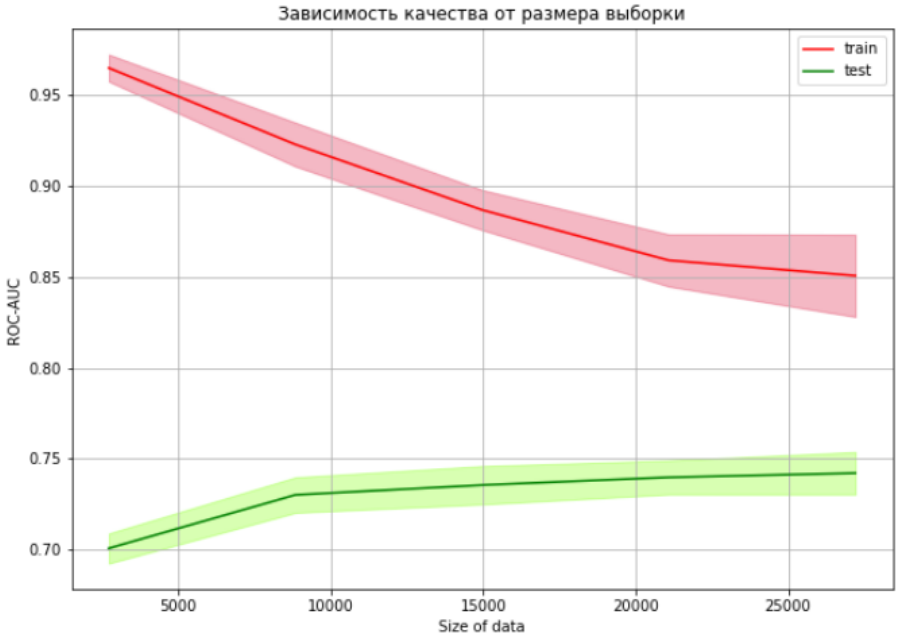
```
pipe = Pipeline([('prep', preprocessor),
                 ('xgbrfc', XGBRFClassifier(n_estimators=120, n_jobs=-1, objective='binary:logistic',
                                            eval_metric='logloss', max_depth=8, reg_lambda=0.1))])
pipe.fit(X_train, y_train)
labels = pipe.predict(X_test)
y_test = pipe.predict_proba(X_test)[:, 1]
test['result'] = y_test
assignment = test[['ID', 'result']]
assignment.to_csv('assignment_tuned_xgbrf.csv', index=False)
```

**Score = 0.70930**

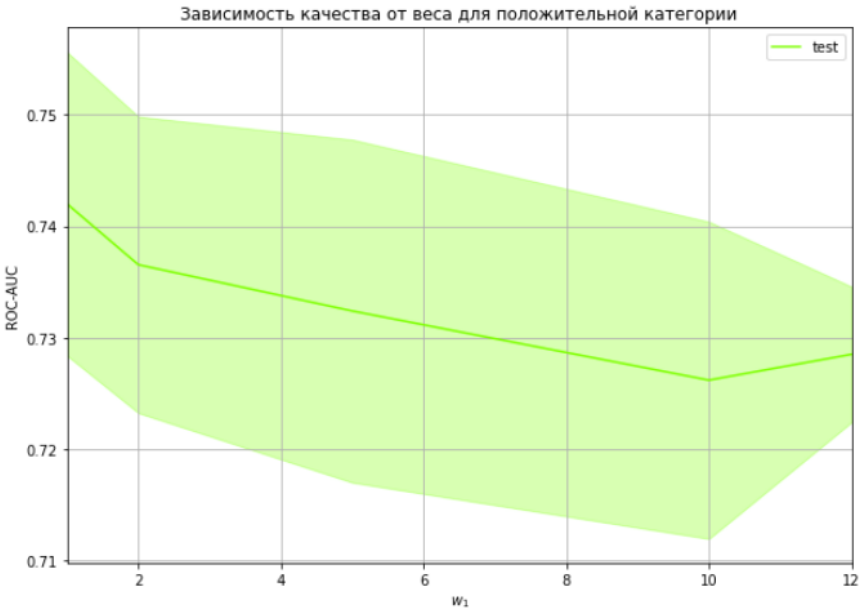
Наша гипотеза о том, что лучший score будет получен с помощью XGBClassifier подтвердилась.
Однако в процессе работы над проектом я встретился с библиотекой CatBoost, которая также является реализацией градиентного бустинга. Её большим преимуществом в данной задаче служит то, что она автоматически обрабатывает категориальные признаки. Также её очень хвалят за лучший score среди других реализаций градиентного бустинга для задач бинарной классификации.
И действительно, данная реализация на kaggle показала наилучший score, равный 0.71901.

# 7 Эксперименты с моделью прогнозирования оттока

**1. Оценка количества объектов для построения качественной модели.**
Для обучения доступна достаточно большая выборка и может так оказаться, что начиная с некоторого момента рост размера обучающей выборки перестает влиять на качество модели.




**2. Разные способы обработки выборки.**
Часто несбалансированные по классам выборки приводят к различным проблемам при обучении моделей, поэтому мы попробуем по-разному обработать выборку, поиграть с распределением объектов по классам и сделать выводы о том, как соотношение классов влияет на качество модели.
**2.1. Задание разных весов объектам.**
Зададим веса объектам так, чтобы соотношение классов с учетом весов объектов изменилось. Первой категории у нас и так много, ей придадим вес 1, будем параметризировать вторую

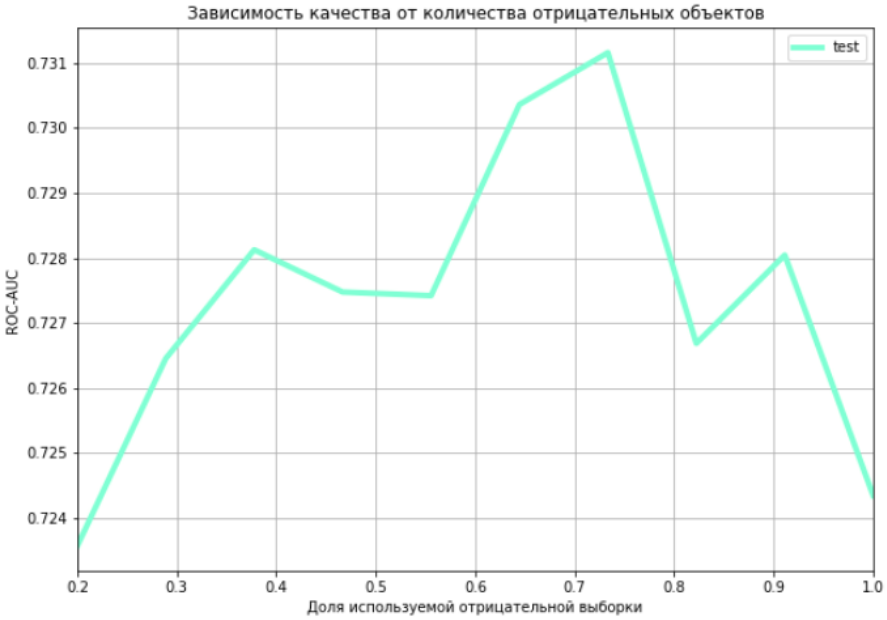



категорию.

На графике видно, что **веса ухудшают ситуацию.**

### 2.2. Undersampling.

Применим к выборке технологию undersampling: для этого нужно убрать из обучения некоторое количество объектов большего класса таким образом, чтобы соотношение классов изменилось.




```
1 max(zip(results, ns))[1]
```

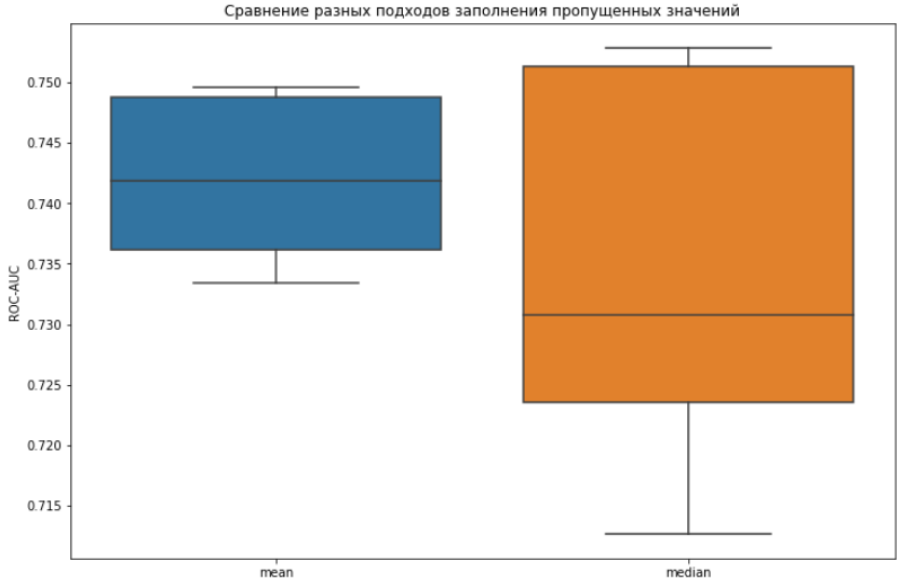
```
0.7333333333333334
```

Результат интересный, андерсемплирование с 3/4 отрицательной выборки может принести интересные результаты. С другой стороны, возможно это всё погрешность. Но протестировать стоит.

### 3. Работа с признаками.

Ранее мы реализовали несколько стратегий для обработки пропущенных значений. Сравним

эти стратегии между собой с помощью оценки качества моделей кросс-валидации, построенных на датасетах с использованием различных стратегий.



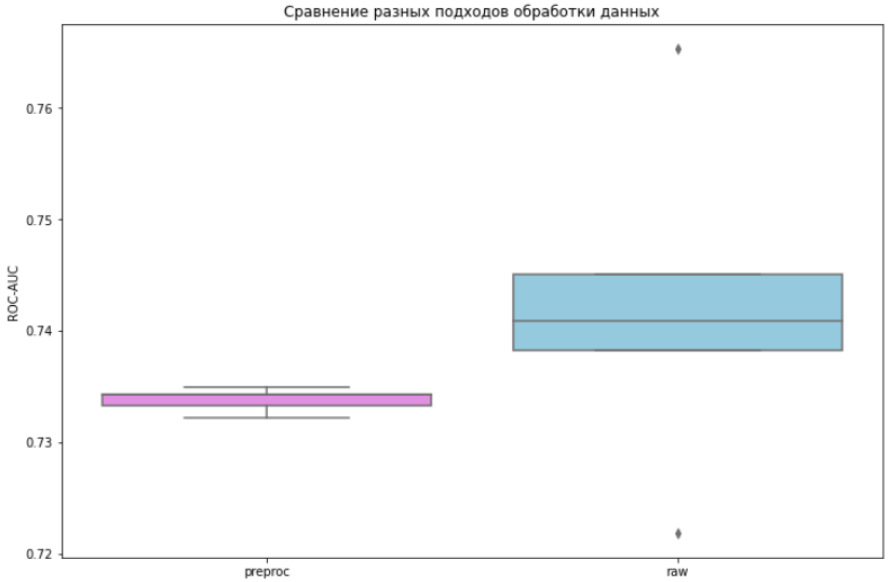

В среднем распределение среднего значения выглядит лучше, мало того, это ещё и подтверждается эмпирически во время отправления разных csv.

**4. Обработка категориальных признаков.**

Напомним очень важное преимущество библиотеки CatBoost: не нужно обрабатывать категориальные значения. Сравним две стратегии между собой:

1. Использовать OneHotEncoder для категориальных признаков с небольшим количеством категорий и BinaryEncoder для остальных. А числовые признаки нормализировать с помощью StandardScaler.
2. Использование CatBoost без обработки категориальных признаков.

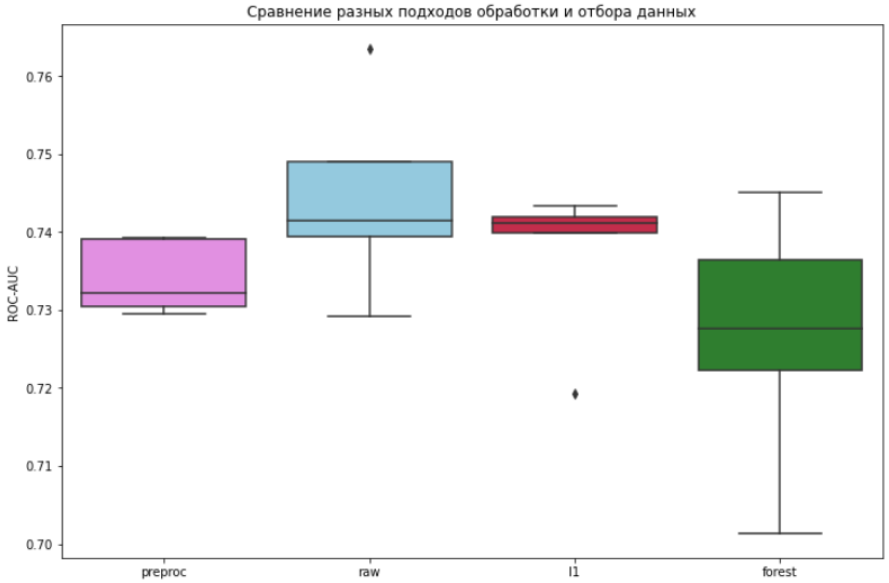
Сравнение этих стратегий между собой производилось с помощью оценки качества моделей по кросс-валидации, построенных на датасетах с использованием различных стратегий.




**CatBoost победил.**

**5. Отбор признаков.**

Отбор признаков в этой задаче является невероятно важным по моему мнению, так как в данных есть много пропусков. Сравним 4 подхода к отбору признаков:




l1-регуляризация получилась довольно интересным вариантом, явно лучше, чем просто обработанный вариант. Но всё ещё хуже, чем датасет без обработки.

**6. Подбор оптимальных параметров модели.**

Есть мнение, что Catboost не требует в принципе тюнинга. И мы в целом убедились, что чаще всего попытки настроить обучение претерпевали поражение. Также обучение на кэтбусте составляет примерно две минуты, что означает, что каждая кросссвалидация обходится в 10 минут. Сама попытка загридсёрчить в таком случае является неэффективной, так как время - тоже ресурс.

**7. Оценка признаков, принесших наибольший вклад в модель.**

Библиотека CatBoost имеет встроенную функцию feature_importances, с помощью которой мы и оценили признаки, которые внесли наибольший и наименьший вклад в модель:

```
1 print('Лучшие пять признаков: ' + ', '.join(top5))
```

```
Лучшие пять признаков: Var126, Var189, Var199, Var192, Var113
```

```
1 print('Худшие пять признаков: ' + ', '.join(antitop))
```

```
Худшие пять признаков: Var2, Var14, Var49, Var62, Var89
```

**8. Просмотр объектов.**

Напоследок посмотрим на объекты, на которых достигаются наибольшие ошибки классификации.

```
top10 = errors.sort_values(ascending=False)[:10]
top10
```

```
32242    0.992717
8218     0.989980
13400    0.989629
8820     0.989233
9077     0.987644
8640     0.987245
21875    0.985928
20197    0.984861
32501    0.983893
21323    0.983822
Name: label, dtype: float64
```

В этих объектах довольно много nanов, поэтому мы попробовали убрать столбцы из nanов. Что привело к довольно сильному сокращение таблицы в несколько раз. То есть значения 135 признаков были предположены нами, когда мы заполняли наны. Мы видим, что у лидирующей шестёрки один из топовых признаков также заполнен нанами.
Вывод: Мы можем работать с нанами и заполнять их, но их влияние может довольно сильно портить всё.

**9. Финальное решение.**

По итогам проведенных экспериментов построим финальную решение - модель с наилучшим качеством.
Для этого мы рассмотрим модель из 6 пункта и модель, построенную на сырых данных, без дополнительной обработки.
К небольшому удивлению, выиграла всё же модель без особой дополнительной обработки. Но необходимо не забывать, что это особенность библиотеки CatBoost.

**10. Как ещё можно улучшить модель?**

Nanы - плохо. Также нужно отобрать признаки. Но при этом ни в коем случае нельзя обрабатывать категориальные признаки. Именно учёт этих выводов помог мне получить самый лучший результат из всех попыток.

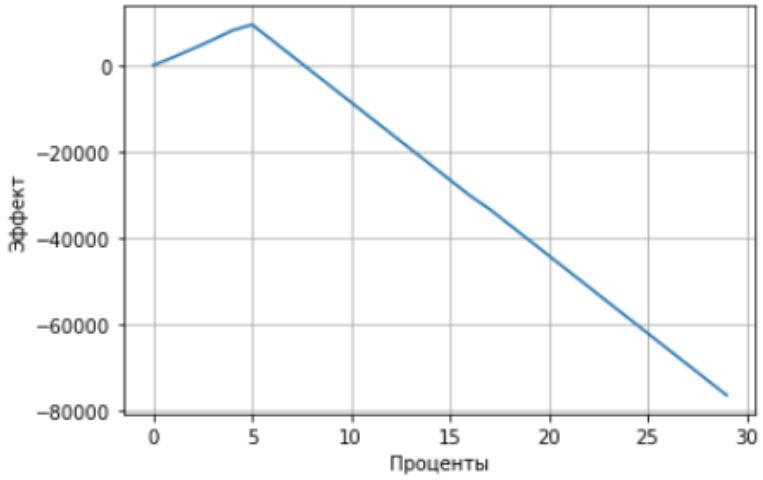
## 8 Оценка эффекта от внедрения полученного решения

На данном этапе нам пригодится наш hold-out датасет, который составил 0,25 от всей обучающей выборки(10 000 записей). Построим экономическую модель и оценим её на этом датасете.
Для начала отберём из этой выборки пользователей, собравшихся уходить, и пользователей, у которых модель определила вероятность ухода $> 0.5$.
Теперь перейдем к построению нашей экономической модели. Для этого введём ряд обозначений:
**FN** - количество пользователей ошибочно отнесенных к "остающимся".
**FP** - количество пользователей ошибочно отнесенных к "уходящим".
**PPM** - количество денег, которые в среднем приносит один пользователь в месяц.
**CFO** - количество денег, которые в среднем вы будете вкладывать в удержание одного пользователя.

**PA** - вероятность принятия пользователем нашего предложения.
**NU** - количетсво пользователей, участвующих в кампании.
**P** - экономический эффект.
Следовательно, получаем формулу, оценивающую экономический эффект от построения модели:

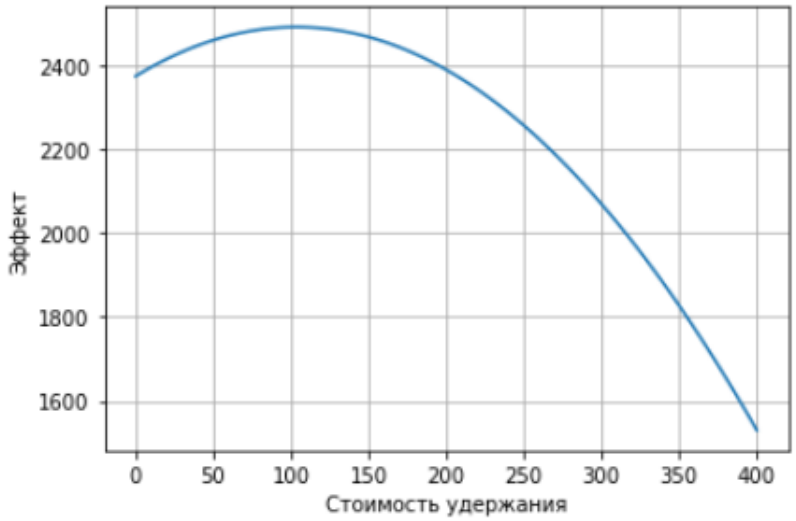
$$P = (NU - FP - FN) * PA * PPM + FP * PPM - CFO * (NU - FN) - FN * PPM * PA$$

Данная модель предполагает, что пользователей склонных к оттоку нужно удерживать каждый месяц.
Пусть PPM = 500 денежных единиц, CFO = 200 ден.ед., PA = 0.9
Эффект в месяц от проведения кампании при участии 1% от отобранных пользователей: 1850.0 ден.ед.
Проанализируем, какой топ пользователей, склонных к оттоку согласно модели, оптимально выбрать для проведения кампании по удержанию для таким образом, чтобы экономический эффект был наибольшим.




**Максимальный экономический эффект достигается при 5 процентах.**
**Количество пользователей: 39**
**Максимальный экономический эффект в месяц: 9450.0 ден.ед.**
Как видно - наибольший экономический эффект наблюдается для выборки из 5% самых склонных к оттоку (по результатам предсказания). Снижение эффекта происходит по причине увеличения количества ошибок угадывания (Так как я использую ранжированный по вероятности оттока список пользователей, после достижения определенного порога (0.5) чем больше выборка - тем больше ошибок).
Усложняем модель. К первоначальной формуле добавим зависимость стоимости удержания пользователя от того - принял он или нет предложение.
CFOY - затраты если принял предложение.
CFON - затраты если отказался.
$P = (NU - FP - FN) * PA * PPM + FP * PPM - CFOY * (NU - FN) * PA - CFON * (NU - -FN) * (1 - PA) - FN * PPM * PA$
Пусть CFON = 100, CFOY = 200.
Варьируем стоимость удержания и вероятность, с которой пользователь принимает предложение. Считаем, что чем дороже обходится удержание, тем выше вероятность принятия

предложения.

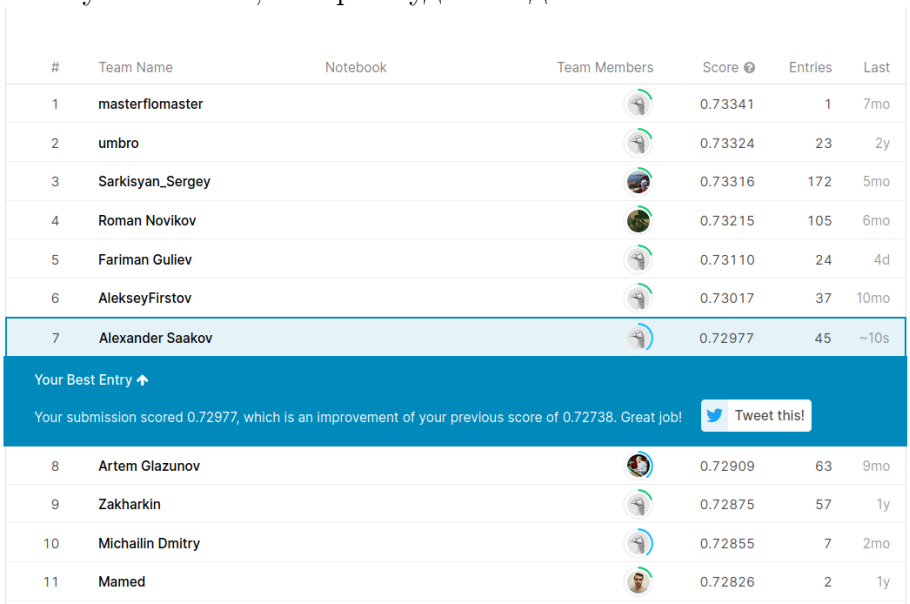



Максимальный экономический эффект достигается при затратах на удержание: 112 ден.ед.
Что соответствует вероятности удержания: 31 %
Максимальный экономический эффект в месяц: 2490 ден. ед.
Затраты на тех, кого не удалось удержать: 29 ден.ед.
Можно сделать вывод, что оптимальной суммой для удержания пользователя является 22,4% от прибыли которую он приносит в месяц.
Оценим изменение экономического эффекта от проведения кампании по удержанию при увеличении качества модели на 1% и на 3%.
В данном случае качество модели определяется количеством ошибок.
FN - модель пропустила пользователя собравшегося уходить. Действия по удержанию не проведены. (большие потери)
FP - проводим действия по удержанию для пользователя ошибочно отнесенного к "уходящим". (небольшие потери)
Соответственно, если качество модели увеличивается на 1-3%, то количество таких ошибок уменьшается.
Если использовать простейшую модель для оценки, то каждая FN это убыток равный упущенной прибыли (PPM * PA) из-за бездействия. Каждая FP - убыток из-за затрат на проведение кампании с пользователем, который и так не собирался уходить (CFO).
*Для улучшения качества модели на 1%:*
Уменьшение количества FN: 7
Уменьшение количества FP: 0
Уменьшение убытков из-за улучшения качества модели на 1%: 1750.0 ден.ед.
*Для улучшения качества модели на 3%:*
Уменьшение количества FN: 22
Уменьшение количества FP: 0
Уменьшение убытков из-за улучшения качества модели на 3%: 5500.0 ден.ед.
**В целом**, вложение средств в реализацию модели может быть оправданным при следующих условиях.
- Более тщательное прогнозирование притока новых пользователей
- Изучение того, насколько часто пользователи, склонные к оттоку, откликаются на акции и предложения о скидке.

Даже незначительное улучшение качества модели приносит доход(например на 1% из пункта 5), но вложение средств в улучшение качества модели является экономически оправданным при условии как минимум равенства затрат на улучшение и прибыли от дополнительно удержанных пользователей.

# 9 Выводы

Лучший score, которого удалось добиться: 0.72977.



| # | Team Name | Notebook | Team Members | Score | Entries | Last |
|---|---|---|---|---|---|---|
| 1 | masterflomaster | | | 0.73341 | 1 | 7mo |
| 2 | umbro | | | 0.73324 | 23 | 2y |
| 3 | Sarkisyan_Sergey | | | 0.73316 | 172 | 5mo |
| 4 | Roman Novikov | | | 0.73215 | 105 | 6mo |
| 5 | Fariman Guliev | | | 0.73110 | 24 | 4d |
| 6 | AlekseyFirstov | | | 0.73017 | 37 | 10mo |
| 7 | Alexander Saakov | | | 0.72977 | 45 | ~10s |
| | Your Best Entry ↑ | | | | | |
| | Your submission scored 0.72977, which is an improvement of your previous score of 0.72738. Great job! Tweet this! | | | | | |
| 8 | Artem Glazunov | | | 0.72909 | 63 | 9mo |
| 9 | Zakharkin | | | 0.72875 | 57 | 1y |
| 10 | Michailin Dmitry | | | 0.72855 | 7 | 2mo |
| 11 | Mamed | | | 0.72826 | 2 | 1y |

Но не смотря на экономическую выгоду модели, необходимо провести ряд тестов/экспериментов для ответа на вопрос, стоит ли внедрять модель в процессы компании. Для этого можно провести A/B-тест, но длительностью не менее 3-х месяцев, в связи с сильным дизбалансом классов. Если провести тест на слишком коротком временном промежутке, то результаты могут стать не достоверными. Так в обязательном порядке нужно делить на группы с учетом дизбаланса классов, чтобы в группах получились те же пропорции классов, что и в исходных данных. Тогда тест будет более приближен к реальным условиям на практике.

# 10 Список Литературы


1. Образовательная платформа Coursera, на курсе которого и был предложен данный проект.
2. Соревновательная платформа kaggle с соревнованием по churn prediction. Режим доступа: https://www.kaggle.com/c/telecom-clients-prediction2
3. Официальная документация библиотеки CatBoost. Режим доступа: https://catboost.ai/
4. Репозиторий github с CatBoost tutorials. Режим доступа: https://github.com/catboost/tutorials