lesson 4

1. скачать набор данных маркетинговых кампаний отсюда https://www.kaggle.com/davinwijaya/customer-retention (https://www.kaggle.com/davinwijaya/customer-retention)
2. там поле conversion - это целевая переменная, а offer - коммуникация. Переименовать поля (conversion -> target, offer -> treatment) и привести поле treatment к бинарному виду (1 или 0, т.е было какое-то предложение или нет) - значение No Offer означает отсутствие коммуникации, а все остальные - наличие.
3. сделать разбиение набора данных на тренировочную и тестовую выборки
4. сделать feature engineering на ваше усмотрение (допускается свобода выбора методов)
5. провести uplift-моделирование 3 способами: одна модель с признаком коммуникации (S learner), модель с трансформацией таргета (трансформация классов п. 2. 1) и вариант с двумя независимыми моделями
6. в конце вывести единую таблицу сравнения метрик uplift@10%, uplift@20% этих 3 моделей
7. построить модель UpliftTreeClassifier и попытаться описать словами полученное дерево
8. (опционально) для модели S learner (модель с дополнительным признаком коммуникации) построить зависимость таргета (конверсии - поле conversion) от значения uplift: 1) сделать прогноз и получить uplift для тестовой выборки 2) отсортировать тестовую выборку по uplift по убыванию 3) разбить на децили (pandas qcut вам в помощь) 4) для каждого дециля посчитать среднюю conversion
9. (опционально) построить модель UpliftRandomForestClassifier и попытаться описать словами полученное дерево

Фичи

- recency - месяцев с момента последней покупки
- history - стоимость покупок клиента в долларах США
- used_discount - использовал ли клиент скидку ранее
- used_bogo - использовал ли клиент услугу "купи и получи" раньше
- zip_code - класс почтового индекса как Пригородный/Городской/Сельский (Suburban/Urban/Rural)
- is_referral - был ли клиент приобретен по реферальному каналу
- channel - каналы, которые использует клиент, Телефон / Интернет / Многоканальный (Phone/Web/Multichannel)

Ввод [1]:
```
import pandas as pd
import numpy as np
from catboost import CatBoostClassifier
import matplotlib.pyplot as plt
```

Ввод [2]:
```
data = pd.read_csv('data.csv', sep=',')
data.head(3)
```

Out[2]:

	recency	history	used_discount	used_bogo	zip_code	is_referral	channel	offer	conversion
0	10	142.44	1	0	Surburban	0	Phone	Buy One Get One	0
1	6	329.08	1	1	Rural	1	Web	No Offer	0

Ввод [3]:
```
data.rename(columns = {'conversion': 'target', 'offer': 'treatment'}, inplace = True)
data.head(3)
```

Out[3]:

	recency	history	used_discount	used_bogo	zip_code	is_referral	channel	treatment	target
0	10	142.44	1	0	Surburban	0	Phone	Buy One Get One	0
1	6	329.08	1	1	Rural	1	Web	No Offer	0
2	7	180.65	0	1	Surburban	1	Web	Buy One Get One	0

Ввод [4]:
```
data.loc[data['treatment']=='No Offer', 'treatment'] = 0
data.loc[data['treatment']!=0, 'treatment'] = 1
data.head(5)
```

Out[4]:

	recency	history	used_discount	used_bogo	zip_code	is_referral	channel	treatment	target
0	10	142.44	1	0	Surburban	0	Phone	1	0
1	6	329.08	1	1	Rural	1	Web	0	0
2	7	180.65	0	1	Surburban	1	Web	1	0
3	9	675.83	1	0	Rural	1	Web	1	0
4	2	45.34	1	0	Urban	0	Web	1	0

Ввод [5]:
```
from sklearn.model_selection import cross_val_score, train_test_split
```

Ввод [6]:
```
data_train, data_test = train_test_split(data, random_state=100500, test_size=0.3)
```

Ввод [7]:
```
data_train.size
```

Out[7]: 403200

Ввод [8]:
```
y_train = data_train['target']
treat_train = data_train['treatment']
X_train = data_train.drop(['target', 'treatment'], axis=1)
```

Ввод [9]:
```
y_train.sum()
```

Out[9]: 6591

Ввод [10]:
```
data_test.size
```

Out[10]: 172800

Ввод [11]:
```
y_test = data_test['target']
treat_test = data_test['treatment']
X_test = data_test.drop(['target', 'treatment'], axis=1)
```

Ввод [12]:
```
y_test.sum()
```

Out[12]: 2803

Ввод [13]:
```
continuos_cols = ['recency', 'history']
cat_cols = ['zip_code', 'channel']
base_cols = [ 'used_discount', 'used_bogo', 'is_referral']
```

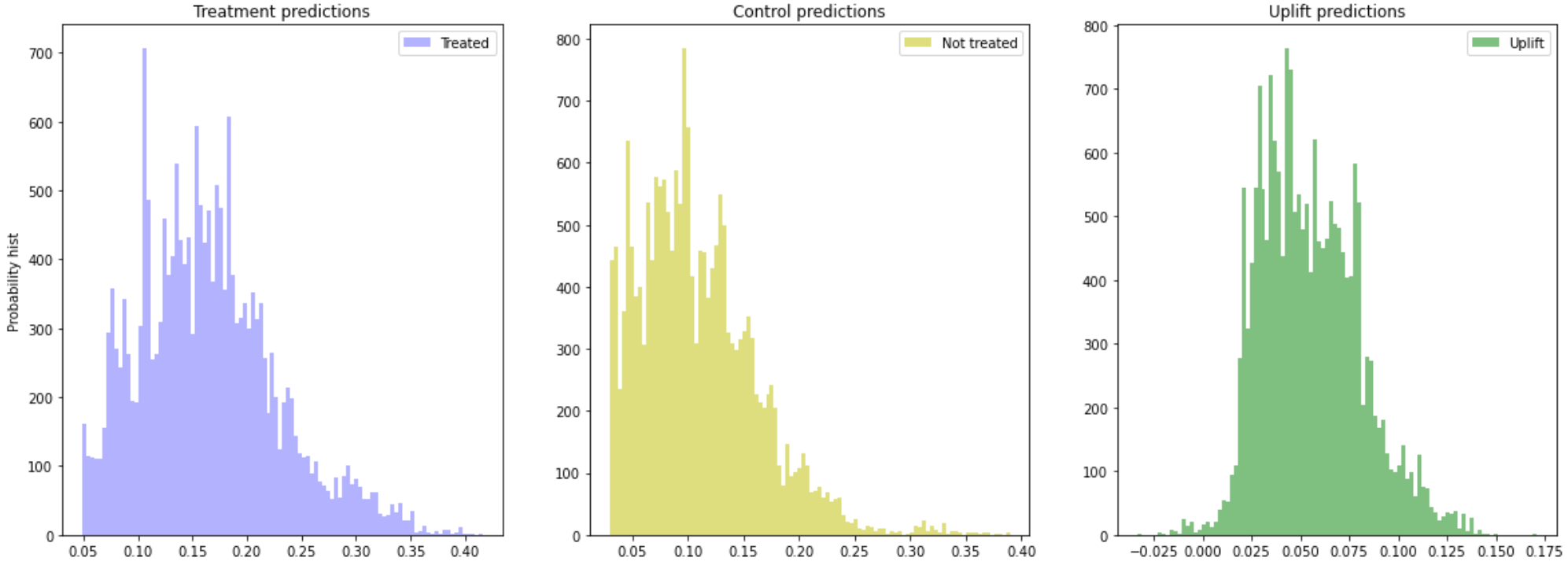
Ввод [14]:
```
from sklift.metrics import uplift_at_k
from sklift.viz import plot_uplift_preds
from sklift.models import SoloModel
```

Ввод [15]:
```
models_results = {
    'approach': [],
    'uplift@10%': [],
    'uplift@20%': []
}
```

Ввод [16]:
```
sm = SoloModel(CatBoostClassifier(iterations=20, thread_count=2, random_state=42, silent=True))
sm = sm.fit(X_train, y_train, treat_train, estimator_fit_params={'cat_features': cat_cols})

uplift_sm = sm.predict(X_test)

sm_score_10 = uplift_at_k(y_true=y_test, uplift=uplift_sm, treatment=treat_test, strategy='by_group', k=0.1)
sm_score_20 = uplift_at_k(y_true=y_test, uplift=uplift_sm, treatment=treat_test, strategy='by_group', k=0.2)

models_results['approach'].append('SoloModel')
models_results['uplift@10%'].append(sm_score_10)
models_results['uplift@20%'].append(sm_score_20)

# Получим условные вероятности выполнения целевого действия при взаимодействии для каждого объекта
sm_trmnt_preds = sm.trmnt_preds_
```

```
# И условные вероятности выполнения целевого действия без взаимодействия для каждого объекта
sm_ctrl_preds = sm.ctrl_preds_

# Отрисуем распределения вероятностей и их разность (uplift)
plot_uplift_preds(trmnt_preds=sm_trmnt_preds, ctrl_preds=sm_ctrl_preds)
```

Out[16]:
```
array([<AxesSubplot:title={'center':'Treatment predictions'}, ylabel='Probability hist'>,
       <AxesSubplot:title={'center':'Control predictions'}>,
       <AxesSubplot:title={'center':'Uplift predictions'}>], dtype=object)
```



Ввод [17]:
```
# С той же легкостью можно обратиться к обученной модели.
# Например, чтобы построить важность признаков:
sm_fi = pd.DataFrame({
    'feature_name': sm.estimator.feature_names_,
    'feature_score': sm.estimator.feature_importances_
}).sort_values('feature_score', ascending=False).reset_index(drop=True)

sm_fi
```

Out[17]:

	feature_name	feature_score
0	is_referral	20.600225
1	treatment	16.953911
2	used_bogo	13.609603

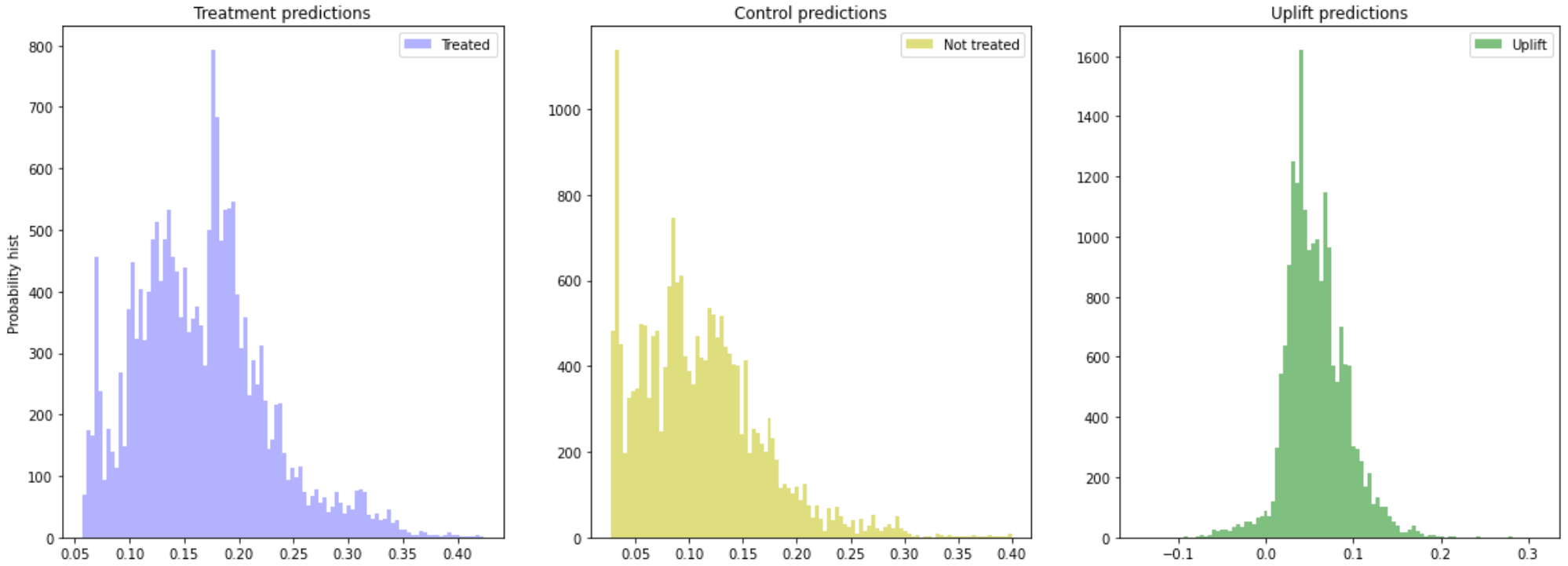
	feature_name	feature_score
3	zip_code	12.631013
4	recency	12.107073
5	history	10.004437
6	used_discount	9.955917

Ввод [18]:
```
from sklift.models import ClassTransformation


ct = ClassTransformation(CatBoostClassifier(iterations=20, thread_count=2, random_state=42, silent=True))
ct = ct.fit(X_train, y_train, treat_train, estimator_fit_params={'cat_features': cat_cols})

uplift_ct = ct.predict(X_test)

ct_score_10 = uplift_at_k(y_true=y_test, uplift=uplift_ct, treatment=treat_test, strategy='by_group', k=0.1)
ct_score_20 = uplift_at_k(y_true=y_test, uplift=uplift_ct, treatment=treat_test, strategy='by_group', k=0.2)

models_results['approach'].append('ClassTransformation')
models_results['uplift@10%'].append(ct_score_10)
models_results['uplift@20%'].append(ct_score_20)
```

```
<ipython-input-18-028461f8726e>:5: UserWarning: It is recommended to use this approach on treatment balanced data. Curre
nt sample size is unbalanced.
  ct = ct.fit(X_train, y_train, treat_train, estimator_fit_params={'cat_features': cat_cols})
```

Ввод [19]:
```
from sklift.models import TwoModels


tm = TwoModels(
    estimator_trmnt=CatBoostClassifier(iterations=20, thread_count=2, random_state=42, silent=True),
    estimator_ctrl=CatBoostClassifier(iterations=20, thread_count=2, random_state=42, silent=True),
    method='vanilla'
)
tm = tm.fit(
    X_train, y_train, treat_train,
    estimator_trmnt_fit_params={'cat_features': cat_cols},
    estimator_ctrl_fit_params={'cat_features': cat_cols}
)

uplift_tm = tm.predict(X_test)

tm_score_10 = uplift_at_k(y_true=y_test, uplift=uplift_tm, treatment=treat_test, strategy='by_group', k=0.1)
tm_score_20 = uplift_at_k(y_true=y_test, uplift=uplift_tm, treatment=treat_test, strategy='by_group', k=0.2)
```

```
models_results['approach'].append('TwoModels')
models_results['uplift@10%'].append(tm_score_10)
models_results['uplift@20%'].append(tm_score_20)

plot_uplift_preds(trmnt_preds=tm.trmnt_preds_, ctrl_preds=tm.ctrl_preds_)
```



Ввод [20]:
```
pd.DataFrame(models_results)
```

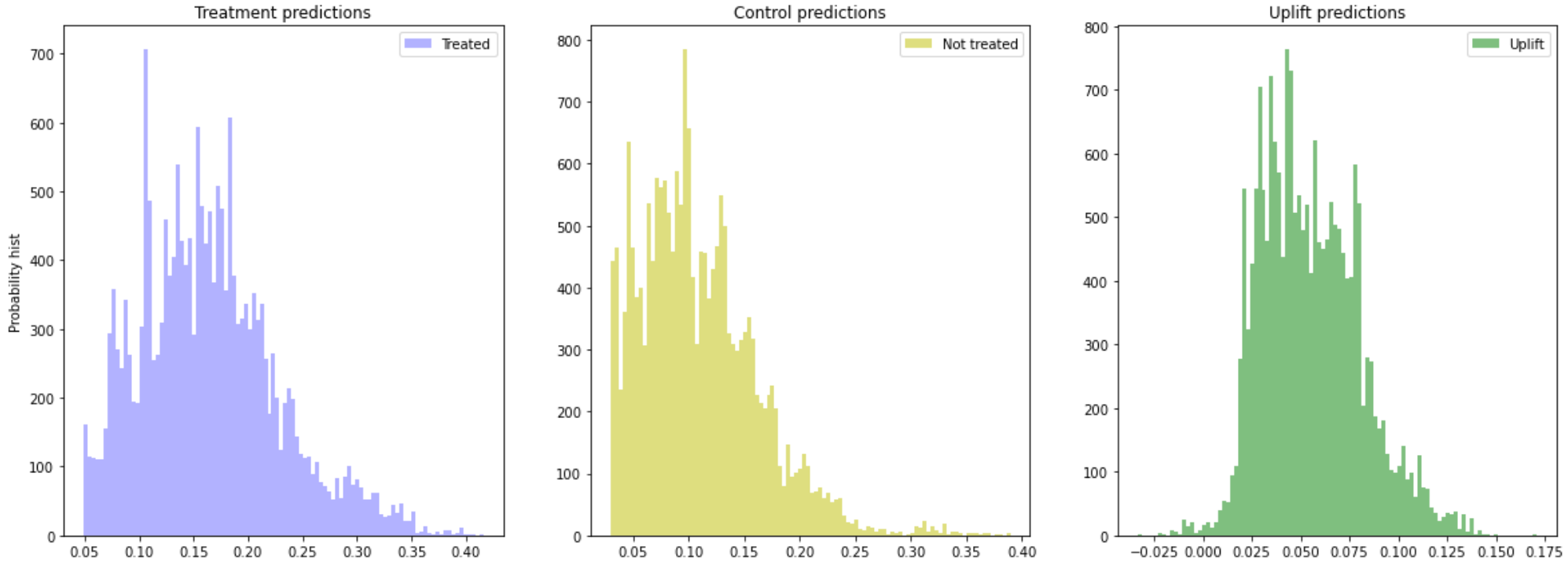
Out[20]:

	approach	uplift@10%	uplift@20%
0	SoloModel	0.101263	0.101850
1	ClassTransformation	0.130983	0.097526
2	TwoModels	0.129540	0.103556

Ввод [21]:
```
sm = SoloModel(CatBoostClassifier(iterations=20, thread_count=2, random_state=42, silent=True))
sm = sm.fit(X_train, y_train, treat_train, estimator_fit_params={'cat_features': cat_cols})

uplift_sm = sm.predict(X_test)

# Получим условные вероятности выполнения целевого действия при взаимодействии для каждого объекта
sm_trmnt_preds = sm.trmnt_preds_
# И условные вероятности выполнения целевого действия без взаимодействия для каждого объекта
sm_ctrl_preds = sm.ctrl_preds_
```

```
sm_uplift = sm.trmnt_preds_ - sm.ctrl_preds_

# Отрисуем распределения вероятностей и их разность (uplift)
plot_uplift_preds(trmnt_preds=sm_trmnt_preds, ctrl_preds=sm_ctrl_preds)
```

Out[21]:
```
array([<AxesSubplot:title={'center':'Treatment predictions'}, ylabel='Probability hist'>,
       <AxesSubplot:title={'center':'Control predictions'}>,
       <AxesSubplot:title={'center':'Uplift predictions'}>], dtype=object)
```



Ввод [22]:
```
tab = pd.DataFrame({'target': y_test,
                    'uplift': sm_uplift})
tab_sort = tab.sort_values(by='uplift', ascending=False)
tab_sort
```

Out[22]:

	target	uplift
823	0	0.171090
53055	1	0.149321
23072	0	0.146130
36852	1	0.143206
46659	1	0.143206

	target	uplift
...	...	...
58988	0	-0.021080
32794	0	-0.021080
25983	0	-0.021341
50915	0	-0.033254
28507	0	-0.033254

Ввод [23]:
```
tab_sort['quantile'] = pd.qcut(tab_sort['uplift'], q=10)
```

Ввод [24]:
```
tab_sort
```

Out[24]:

	target	uplift	quantile
823	0	0.171090	(0.0878, 0.171]
53055	1	0.149321	(0.0878, 0.171]
23072	0	0.146130	(0.0878, 0.171]
36852	1	0.143206	(0.0878, 0.171]
46659	1	0.143206	(0.0878, 0.171]
...	...	...	...
58988	0	-0.021080	(-0.034300000000000004, 0.0251]
32794	0	-0.021080	(-0.034300000000000004, 0.0251]
25983	0	-0.021341	(-0.034300000000000004, 0.0251]
50915	0	-0.033254	(-0.034300000000000004, 0.0251]
28507	0	-0.033254	(-0.034300000000000004, 0.0251]

19200 rows × 3 columns

Ввод [25]:
```
result_tab = tab_sort.groupby('quantile').agg({'target': 'mean',
                                               'uplift': 'mean'})
result = np.array(result_tab)
result_tab
```

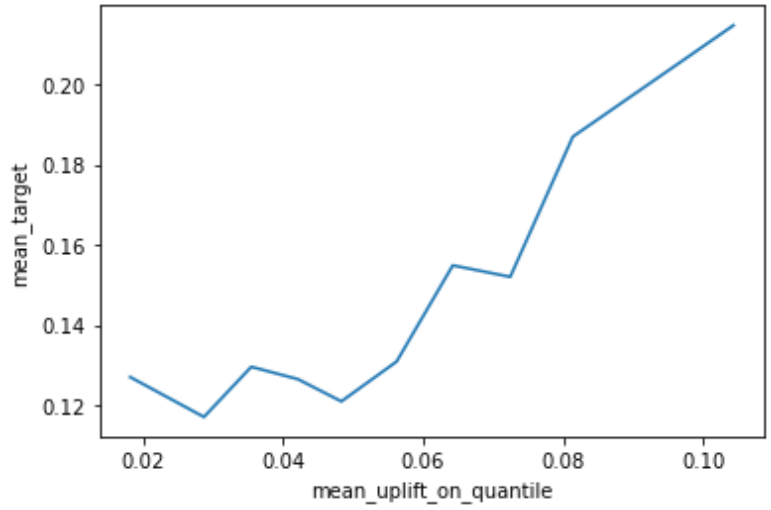
Out[25]:

quantile	target	uplift

quantile	target	uplift
(-0.034300000000000004, 0.0251]	0.127083	0.018119
(0.0251, 0.0321]	0.117159	0.028658
(0.0321, 0.0387]	0.129639	0.035428
(0.0387, 0.0449]	0.126562	0.042096
(0.0449, 0.0521]	0.121039	0.048302
(0.0521, 0.0599]	0.130946	0.056196
(0.0599, 0.068]	0.154870	0.064228
(0.068, 0.0771]	0.152017	0.072439

Ввод [26]:
```
plt.plot(result[:, 1], result[:, 0])
plt.xlabel('mean_uplift_on_quantile')
plt.ylabel('mean_target')
```

Out[26]:
```
Text(0, 0.5, 'mean_target')
```



Ввод [ ]: