Машинное обучение в бизнесе

Урок 4. Uplift-моделирование (моделирование роста)

Домашнее задание

1. Скачать набор данных маркетинговых кампаний отсюда https://www.kaggle.com/davinwijaya/customer-retention (https://www.kaggle.com/davinwijaya/customer-retention)
2. 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. (опционально) 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

Ссылки

https://towardsdatascience.com/a-quick-uplift-modeling-introduction-6e14de32bfe0 (https://towardsdatascience.com/a-quick-uplift-modeling-introduction-6e14de32bfe0)
https://habr.com/ru/company/ru_mts/blog/485980/#reference1 (https://habr.com/ru/company/ru_mts/blog/485980/#reference1)
https://en.wikipedia.org/wiki/Uplift_modelling (https://en.wikipedia.org/wiki/Uplift_modelling)
https://www.youtube.com/watch?v=yFQAIJBYXI0 (https://www.youtube.com/watch?v=yFQAIJBYXI0)
https://www.youtube.com/watch?v=jCUcYiBK03I (https://www.youtube.com/watch?v=jCUcYiBK03I)
https://www.uplift-modeling.com/en/latest/ (https://www.uplift-modeling.com/en/latest/)
https://arxiv.org/pdf/1809.04559.pdf (https://arxiv.org/pdf/1809.04559.pdf)
https://catboost.ai/docs/concepts/about.html (https://catboost.ai/docs/concepts/about.html)

Библиотеки и пакеты
causalml
sklift
catboost

Словарь:

1. uplift

- подъем (rise, lifting, lift, climb, ascent, uplift)
- поднимать (lift, raise, up, pick up, put up, uplift)

2.treatment

- лечение (treatment, therapy, medication, cure, healing, curing)
- обработка (processing, treatment, handling, working, work, cultivation)

Практическое задание

1 Задание

Скачать набор данных маркетинговых кампаний отсюда https://www.kaggle.com/davinwijaya/customer-retention (https://www.kaggle.com/davinwijaya/customer-retention)

Data Exploration

B [1]:
```
import numpy as np, matplotlib as mpl, matplotlib.pyplot as plt, pandas as pd
```

B [2]:
```
# Import data
# df_data = pd.read_csv('/kaggle/input/customer-retention/data.csv')
df_data = pd.read_csv('data.csv')
df_model = df_data.copy()
```

B [3]:
```
# Let's take a look at our data
df_model.head(5)
```

Out[3]:

	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
2	7	180.65	0	1	Surburban	1	Web	Buy One Get One	0
3	9	675.83	1	0	Rural	1	Web	Discount	0
4	2	45.34	1	0	Urban	0	Web	Buy One Get One	0

B [4]:
```
df_model.columns
```

Out[4]:
```
Index(['recency', 'history', 'used_discount', 'used_bogo', 'zip_code',
       'is_referral', 'channel', 'offer', 'conversion'],
      dtype='object')
```

B [5]:
```
# Checking for null data
df_model.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64000 entries, 0 to 63999
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   recency        64000 non-null  int64
 1   history        64000 non-null  float64
 2   used_discount  64000 non-null  int64
 3   used_bogo      64000 non-null  int64
 4   zip_code       64000 non-null  object
 5   is_referral    64000 non-null  int64
 6   channel        64000 non-null  object
 7   offer          64000 non-null  object
 8   conversion     64000 non-null  int64
dtypes: float64(1), int64(5), object(3)
memory usage: 4.4+ MB
```

B [6]:
```
df_model.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
recency	64000.0	5.763734	3.507592	1.00	2.00	6.00	9.0000	12.00
history	64000.0	242.085656	256.158608	29.99	64.66	158.11	325.6575	3345.93
used_discount	64000.0	0.551031	0.497393	0.00	0.00	1.00	1.0000	1.00
used_bogo	64000.0	0.549719	0.497526	0.00	0.00	1.00	1.0000	1.00
is_referral	64000.0	0.502250	0.499999	0.00	0.00	1.00	1.0000	1.00
conversion	64000.0	0.146781	0.353890	0.00	0.00	0.00	0.0000	1.00

There is no null data in the dataset.

B [7]:
```
# Checking for object data
df_model.describe(include=np.object).T
```

Out[7]:

	count	unique	top	freq
zip_code	64000	3	Surburban	28776
channel	64000	3	Web	28217
offer	64000	3	Buy One Get One	21387

B [8]:
```
# Checking unique object data
object_cols = [col for col in df_model.columns if df_model[col].dtype == "object"]
for obj in object_cols:
    print(f'\n{obj}')
    for unique in df_model[obj].unique():
        print(f'- {unique} {sum(df_model[obj] == unique)}')

# for obj in object_cols:
#     print('----')
#     print(df_model[obj].value_counts())
```

```
zip_code
- Surburban 28776
- Rural 9563
- Urban 25661

channel
- Phone 28021
- Web 28217
- Multichannel 7762

offer
- Buy One Get One 21387
- No Offer 21306
- Discount 21307
```

B [9]:
```
#train_num_features = train.select_dtypes(include=['object'])

#object_cols.hist(figsize=(16, 16), bins=50, grid=True)
```

2 Задание

Поле conversion - это целевая переменная, а offer - коммуникация.

- переименовать поля (conversion -> target, offer -> treatment)
- 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.

Data Preprocessing

B [10]:
```
# Переименовать поля:
# conversion -> target
df_model = df_model.rename(columns={'conversion': 'target'})
# offer -> treatment
df_model = df_model.rename(columns={'offer': 'treatment'})

# Приводим поле treatment к бинарному виду (1 или 0, т.е было какое-то предложение или нет)
df_model.treatment = df_model.treatment.map({'No Offer': 0, 'Buy One Get One': 1, 'Discount': 1})
# df_model.treatment = df_model.treatment.map({'No Offer': 0, 'Buy One Get One': -1, 'Discount': 1})
```

B [11]:
```
print(df_model.columns)
df_model.head(6)
```

```
Index(['recency', 'history', 'used_discount', 'used_bogo', 'zip_code',
       'is_referral', 'channel', 'treatment', 'target'],
      dtype='object')
```

Out[11]:

	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
5	6	134.83	0	1	Surburban	0	Phone	1	1

B [12]:
```
# Checking unique object data
object_cols = [col for col in df_model.columns if (col == "treatment") | (col == "target")]
for obj in object_cols:
    print(f'\n{obj}')
    for unique in df_model[obj].unique():
        print(f'- {unique} {sum(df_model[obj] == unique)}')
```

```
treatment
- 1 42694
- 0 21306

target
- 0 54606
- 1 9394
```

B [13]:
```
df_model.groupby("treatment")['target'].describe()
```

Out[13]:

treatment	count	mean	std	min	25%	50%	75%	max
0	21306.0	0.106167	0.308059	0.0	0.0	0.0	0.0	1.0
1	42694.0	0.167049	0.373024	0.0	0.0	0.0	0.0	1.0

B [14]:
```
# pd.Timestamp('1970-01-01'), pd.Timedelta('1s')
```

3. Задание

Сделать разбиение набора данных на тренировочную и тестовую выборки

B [15]:
```
from sklearn.model_selection import train_test_split
```

B [16]:
```
df_model.describe().T
```

Out[16]:

	count	mean	std	min	25%	50%	75%	max
recency	64000.0	5.763734	3.507592	1.00	2.00	6.00	9.0000	12.00
history	64000.0	242.085656	256.158608	29.99	64.66	158.11	325.6575	3345.93
used_discount	64000.0	0.551031	0.497393	0.00	0.00	1.00	1.0000	1.00
used_bogo	64000.0	0.549719	0.497526	0.00	0.00	1.00	1.0000	1.00
is_referral	64000.0	0.502250	0.499999	0.00	0.00	1.00	1.0000	1.00
treatment	64000.0	0.667094	0.471257	0.00	0.00	1.00	1.0000	1.00
target	64000.0	0.146781	0.353890	0.00	0.00	0.00	0.0000	1.00

B [17]:
```
df_model.columns
```

Out[17]:
```
Index(['recency', 'history', 'used_discount', 'used_bogo', 'zip_code',
       'is_referral', 'channel', 'treatment', 'target'],
      dtype='object')
```

B [18]:
```
X_features = ['recency', 'history', 'used_discount', 'used_bogo', 'zip_code', 'is_referral','channel']

y_featutes = ['treatment', 'target']

# категориальные признаки
cat_features = ['zip_code', 'channel']
```

B [19]:
```
# Извлечение признаков
df_features = df_model[X_features].copy()
df_features.head(3)
```

Out[19]:

	recency	history	used_discount	used_bogo	zip_code	is_referral	channel
0	10	142.44	1	0	Surburban	0	Phone
1	6	329.08	1	1	Rural	1	Web
2	7	180.65	0	1	Surburban	1	Web

B [20]:
```
df_train = df_model[y_featutes].copy()
df_train.head(3)
```

Out[20]:

	treatment	target
0	1	0
1	0	0
2	1	0

B [21]:
```
indices_learn, indices_valid = train_test_split(df_train.index, test_size=0.3, random_state=123)

X_train = df_features.loc[indices_learn, :]
y_train = df_train.loc[indices_learn, 'target']
treat_train = df_train.loc[indices_learn, 'treatment']

X_val = df_features.loc[indices_valid, :]
y_val = df_train.loc[indices_valid, 'target']
treat_val =  df_train.loc[indices_valid, 'treatment']
```

B [22]:
```
# # Первое разбиение (10%)
# indices_learn, indices_valid = train_test_split(df_train.index, test_size=0.1, random_state=123)

# X_train_10 = df_features.loc[indices_learn, :]
# y_train_10 = df_train.loc[indices_learn, 'target']
# treat_train_10 = df_train.loc[indices_learn, 'treatment']

# X_val_10 = df_features.loc[indices_valid, :]
# y_val_10 = df_train.loc[indices_valid, 'target']
# treat_val_10 =  df_train.loc[indices_valid, 'treatment']
```

B [23]:
```
# # Второе разбиение (20%)
# indices_learn, indices_valid = train_test_split(df_train.index, test_size=0.2, random_state=123)

# X_train_20 = df_features.loc[indices_learn, :]
# y_train_20 = df_train.loc[indices_learn, 'target']
# treat_train_20 = df_train.loc[indices_learn, 'treatment']

# X_val_20 = df_features.loc[indices_valid, :]
# y_val_20 = df_train.loc[indices_valid, 'target']
# treat_val_20 =  df_train.loc[indices_valid, 'treatment']
```

B [24]:
```
models_results = {
    'approach': [],
    'uplift@10%': [],
    'uplift@20%': []
}
```

B [25]:
```
models_results
```

Out[25]:
```
{'approach': [], 'uplift@10%': [], 'uplift@20%': []}
```

4. Задание

Сделать feature engineering на ваше усмотрение (допускается свобода выбора методов)

B [26]:
```
# One-Hot Encoding:
df_model = pd.get_dummies(df_model)
```

B [27]:
```
df_model.head()
```

Out[27]:

	recency	history	used_discount	used_bogo	is_referral	treatment	target	zip_code_Rural	zip_code_Surburban	zip_code_Urban	channel_Multich
0	10	142.44	1	0	0	1	0	0	1	0	
1	6	329.08	1	1	1	0	0	1	0	0	
2	7	180.65	0	1	1	1	0	0	1	0	
3	9	675.83	1	0	1	1	0	1	0	0	
4	2	45.34	1	0	0	1	0	0	0	1	

5. Задание

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

В [28]:
```
# !pip install scikit-uplift==0.2.0
```

В [29]:
```
# Инструкция по установке пакета: https://github.com/maks-sh/scikit-uplift
# Ссылка на документацию: https://scikit-uplift.readthedocs.io/en/latest/
from sklift.metrics import uplift_at_k
from sklift.viz import plot_uplift_preds
from sklift.models import SoloModel

# sklift поддерживает любые модели,
# которые удовлетворяют соглашениями scikit-learn
# Для примера воспользуемся catboost
from catboost import CatBoostClassifier
```

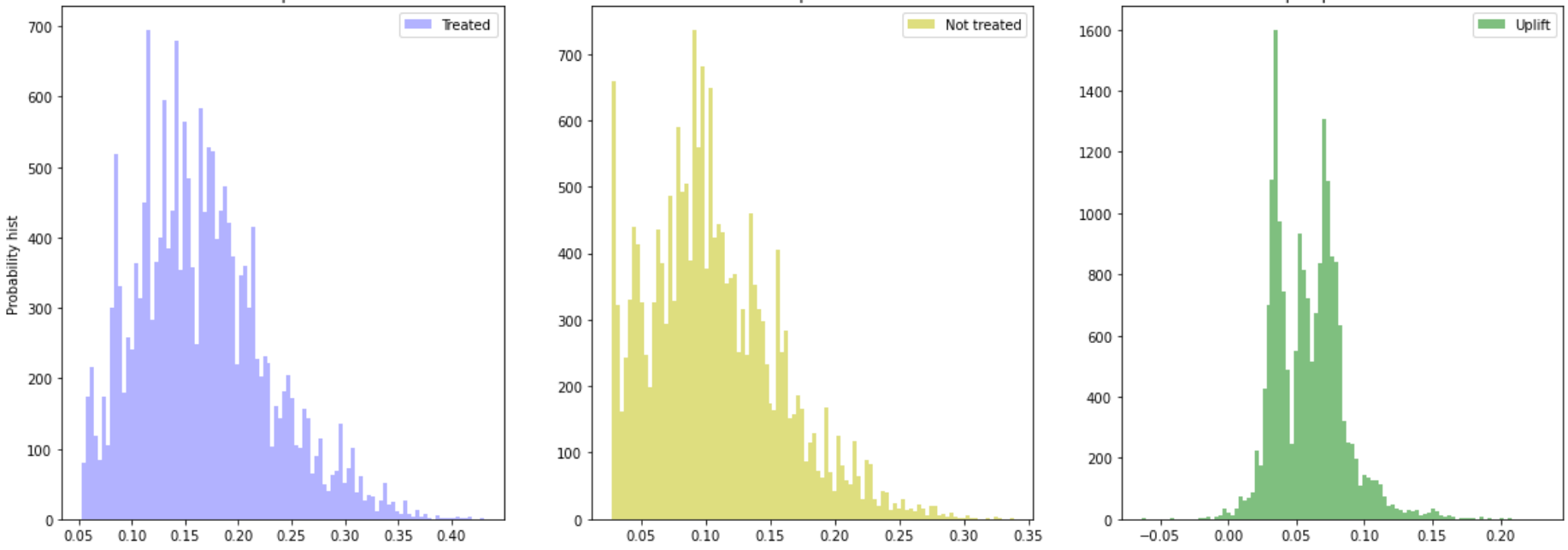
5.1 Одна модель с признаком коммуникации (S learner)

В [30]:
```
# def solo_Model(uplift_percent, X_train, y_train, treat_train, X_val, y_val, treat_val):
def solo_Model(uplift_percent, percent):
    sm = SoloModel(CatBoostClassifier(iterations=20, thread_count=2, random_state=42, silent=True))

    # cat_features-категориальные признаки
    sm = sm.fit(X_train, y_train, treat_train, estimator_fit_params={'cat_features': cat_features})

    uplift_sm = sm.predict(X_val)

    sm_score = uplift_at_k(y_true=y_val, uplift=uplift_sm, treatment=treat_val, strategy='by_group', k=0.3)
    sm_score = uplift_at_k(y_true=y_val, uplift=uplift_sm, treatment=treat_val, strategy='by_group', k=percent)
    # print(f'uplift@10%: {sm_score:.4f}')
    print(f'{uplift_percent}: {sm_score:.4f}\n')

    # models_results['approach'].append('SoloModel')
    # models_results['uplift@10%'].append(sm_score)
    models_results[uplift_percent].append(sm_score)

    # 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);

    print('Посмотрим на топ-признаки:\n')
    # 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)

    print(sm_fi)
```

B [31]:
```
models_results['approach'].append('SoloModel')
percent = 0.1
uplift_percent = 'uplift@10%'
# solo_Model(uplift_percent, X_train_10, y_train_10, treat_train_10, X_val_10, y_val_10, treat_val_10)
solo_Model(uplift_percent, percent)
models_results
```

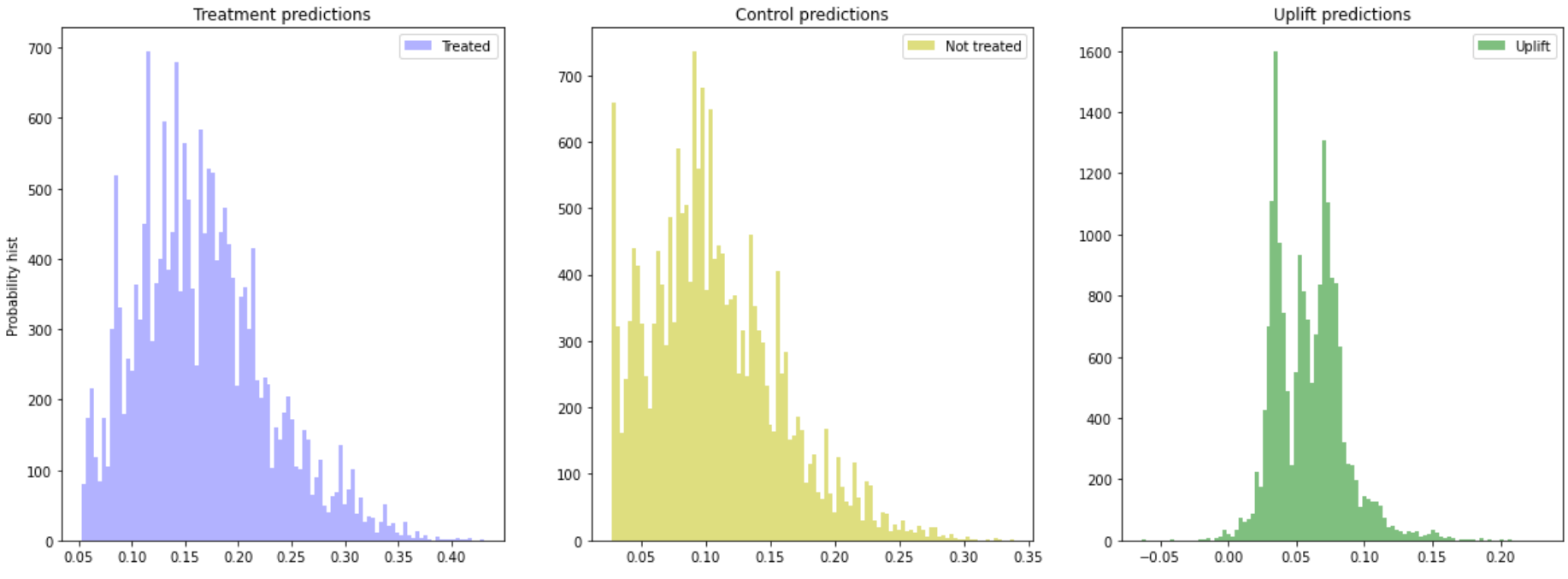
```
uplift@10%: 0.0893

Посмотрим на топ-признаки:

    feature_name  feature_score
0    is_referral      19.189426
1      treatment      17.927079
2      used_bogo      12.651803
3        recency      11.793690
4        channel      11.197806
5       zip_code       9.776886
6  used_discount       8.943366
7        history       8.519945
```

Out[31]:
```
{'approach': ['SoloModel'],
 'uplift@10%': [0.08925430023455824],
 'uplift@20%': []}
```



B [32]:
```
percent = 0.2
uplift_percent = 'uplift@20%'
# solo_Model(uplift_percent, X_train_20, y_train_20, treat_train_20, X_val_20, y_val_20, treat_val_20)
solo_Model(uplift_percent, percent)
# models_results
```

```
uplift@20%: 0.0790

Посмотрим на топ-признаки:

    feature_name  feature_score
0    is_referral      19.189426
1      treatment      17.927079
2      used_bogo      12.651803
3        recency      11.793690
4        channel      11.197806
5       zip_code       9.776886
6  used_discount       8.943366
7        history       8.519945
```



B [33]:
```
pd.DataFrame(data=models_results).sort_values('uplift@10%', ascending=False)
```

Out[33]:

	approach	uplift@10%	uplift@20%
0	SoloModel	0.089254	0.078994

5.2 Модель с трансформацией таргета (трансформация классов п. 2. 1)

B [34]:
```
from sklift.models import ClassTransformation
```

B [35]:
```
models_results['approach'].append('ClassTransformation')
```

B [36]:
```
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_features})

uplift_ct = ct.predict(X_val)

# 10%
ct_score = uplift_at_k(y_true=y_val, uplift=uplift_ct, treatment=treat_val, strategy='by_group', k=0.1)

models_results['uplift@10%'].append(ct_score)
```

```
<ipython-input-36-f2ccf997c437>:2: UserWarning: It is recommended to use this approach on treatment balanced data. Curr
ent sample size is unbalanced.
  ct = ct.fit(X_train, y_train, treat_train, estimator_fit_params={'cat_features': cat_features})
```

B [37]:
```
# 20%
ct_score = uplift_at_k(y_true=y_val, uplift=uplift_ct, treatment=treat_val, strategy='by_group', k=0.2)

models_results['uplift@20%'].append(ct_score)
```

B [38]:
```
pd.DataFrame(data=models_results).sort_values('uplift@10%', ascending=False)
```

Out[38]:

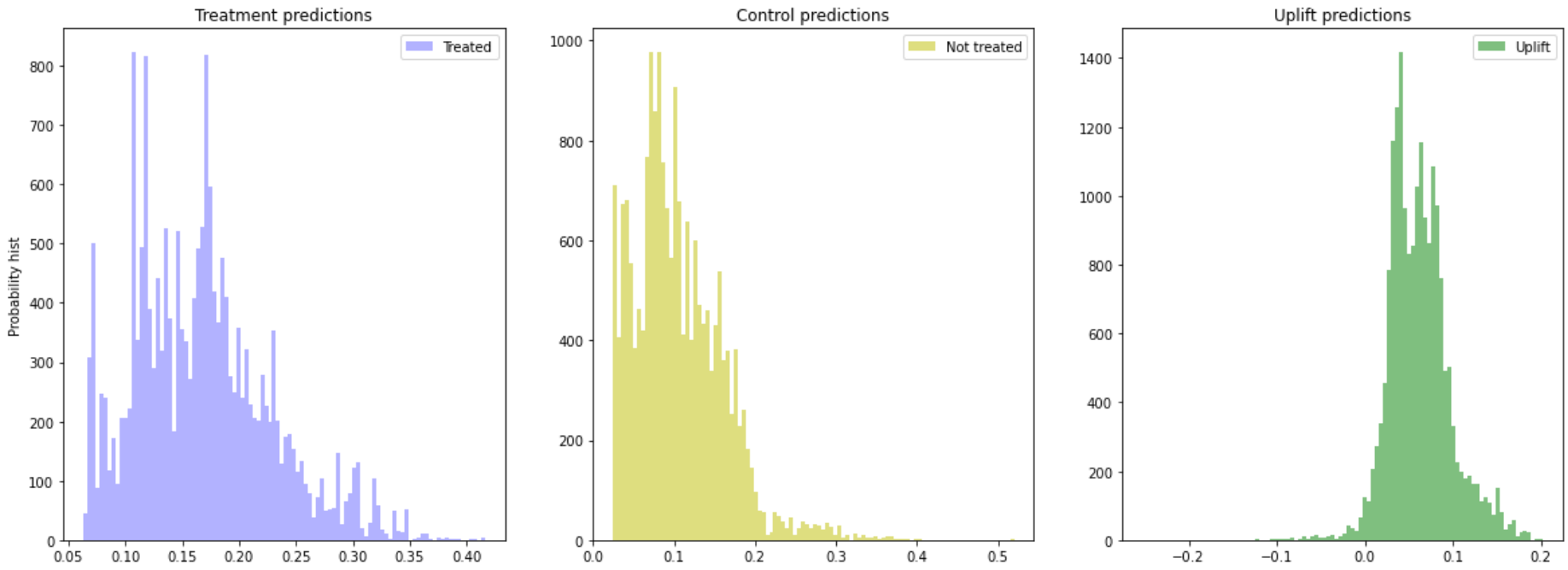
	approach	uplift@10%	uplift@20%
1	ClassTransformation	0.117411	0.095800
0	SoloModel	0.089254	0.078994

5.3 Две независимые модели

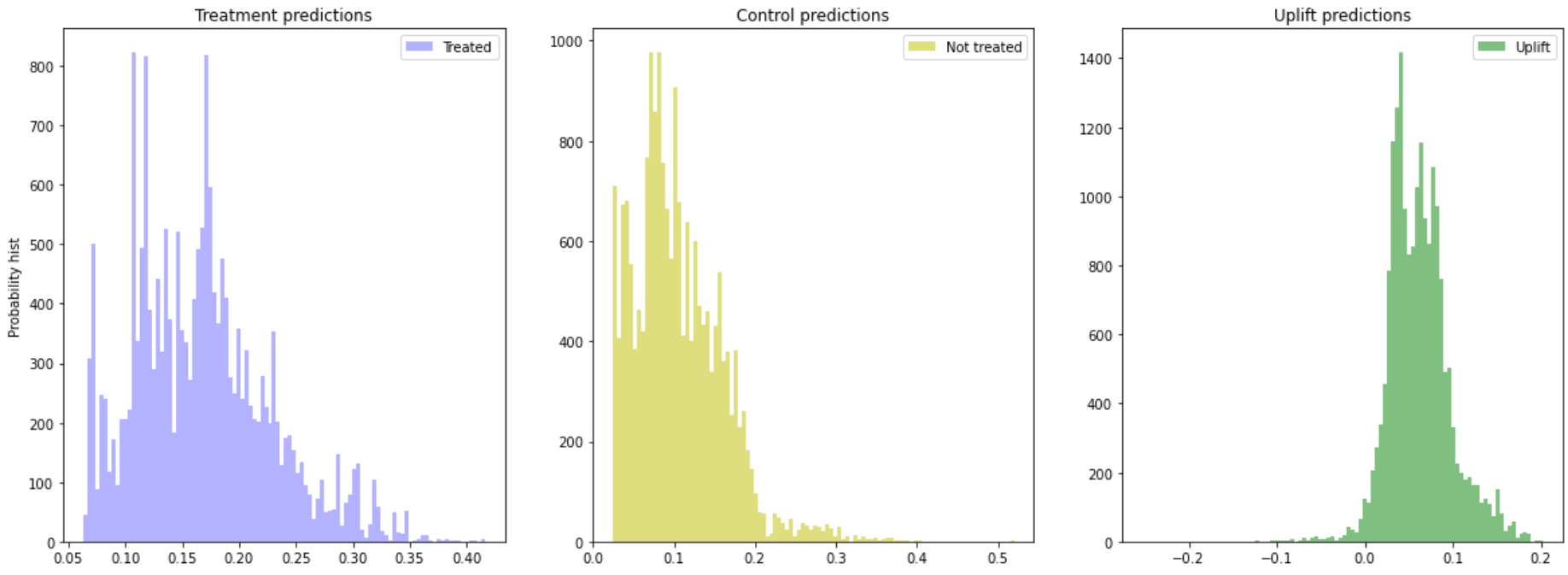
B [39]:
```
from sklift.models import TwoModels
```

B [40]:
```
models_results['approach'].append('TwoModels')
```

B [41]:
```
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_features},
    estimator_ctrl_fit_params={'cat_features': cat_features}
)

uplift_tm = tm.predict(X_val)

# 10%
tm_score = uplift_at_k(y_true=y_val, uplift=uplift_tm, treatment=treat_val, strategy='by_group', k=0.1)

models_results['uplift@10%'].append(tm_score)

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



B [42]:
```
# 20%
tm_score = uplift_at_k(y_true=y_val, uplift=uplift_tm, treatment=treat_val, strategy='by_group', k=0.2)

models_results['uplift@20%'].append(tm_score)

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



Посмотрим на результаты

B [43]:
```
pd.DataFrame(data=models_results).sort_values('uplift@10%', ascending=False)
```

Out[43]:

	approach	uplift@10%	uplift@20%
1	ClassTransformation	0.117411	0.095800
0	SoloModel	0.089254	0.078994
2	TwoModels	0.078306	0.068439

6. Задание

В конце вывести единую таблицу сравнения метрик uplift@10%, uplift@20% этих 3 моделей

B [44]:
```
pd.DataFrame(data=models_results).sort_values('uplift@10%', ascending=False)
```

Out[44]:

	approach	uplift@10%	uplift@20%
1	ClassTransformation	0.117411	0.095800
0	SoloModel	0.089254	0.078994
2	TwoModels	0.078306	0.068439

7. (опционально) Задание

построить модель UpliftTreeClassifier и попытаться описать словами полученное дерево

Вариант с деревом решений

B [45]:
```
# категориальные признаки
cat_features = ['zip_code', 'channel']

X_train_tree = pd.concat([X_train.drop(cat_features, 1),
                          pd.get_dummies(X_train[cat_features], prefix=cat_features)], 1)
features = [col for col in X_train_tree]
```

B [46]:
```
features
```

Out[46]:
```
['recency',
 'history',
 'used_discount',
 'used_bogo',
 'is_referral',
 'zip_code_Rural',
 'zip_code_Surburban',
 'zip_code_Urban',
 'channel_Multichannel',
 'channel_Phone',
 'channel_Web']
```

B [64]:
```
# #!git clone https://github.com/ub8er/causalml.git
# !git clone https://github.com/uber/causalml.git

%cd C:\Install\1\causalml
# !pip install -r requirements.txt
!python setup.py build_ext --inplace
```

```
C:\Install\1\causalml
running build_ext
building 'causalml.inference.tree.causaltree' extension

error: Microsoft Visual C++ 14.0 or greater is required. Get it with "Microsoft C++ Build Tools": https://visualstudio.microsoft.com/visual-cpp-build-tools/ (https://visualstudio.microsoft.com/visual-cpp-build-tools/)
```

B [60]:
```
!python setup.py install
```

```
running install
running bdist_egg
running egg_info
writing causalml.egg-info\PKG-INFO
writing dependency_links to causalml.egg-info\dependency_links.txt
writing requirements to causalml.egg-info\requires.txt
writing top-level names to causalml.egg-info\top_level.txt
reading manifest file 'causalml.egg-info\SOURCES.txt'
reading manifest template 'MANIFEST.in'
writing manifest file 'causalml.egg-info\SOURCES.txt'
installing library code to build\bdist.win-amd64\egg
running install_lib
running build_py
running build_ext
building 'causalml.inference.tree.causaltree' extension

warning: no files found matching '*.pxd' under directory 'causalml'
warning: no files found matching '*.h' under directory 'causalml'
error: Microsoft Visual C++ 14.0 or greater is required. Get it with "Microsoft C++ Build Tools": https://visualstudio.microsoft.com/visual-cpp-build-tools/ (https://visualstudio.microsoft.com/visual-cpp-build-tools/)
```

B [61]:
```
# # You can manually add the virtual environment to Jupyter notebook as a "kernel" as follows:
# conda activate causalml-py38
# pip install ipykernel
# python -m ipykernel install --user --name causalml-py38 --display-name "Python 3.8 (causalml)"
```

B [62]:
```
%%time
from IPython.display import Image
from causalml.inference.tree import UpliftTreeClassifier, UpliftRandomForestClassifier
from causalml.inference.tree import uplift_tree_string, uplift_tree_plot

uplift_model = UpliftTreeClassifier(max_depth=8, min_samples_leaf=200, min_samples_treatment=50,
                                    n_reg=100, evaluationFunction='KL', control_name='control')

uplift_model.fit(X_train_tree.values,
                 treatment=treat_train.map({1: 'treatment', 0: 'control'}).values,
                 y=y_train)

graph = uplift_tree_plot(uplift_model.fitted_uplift_tree, features)
Image(graph.create_png())
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<timed exec> in <module>

ModuleNotFoundError: No module named 'causalml.inference'
```

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

B [ ]:

B [ ]: