GeekBrains, ML in Business

Lesson 6 Homework

Импорт библиотек

In [1]:
```
import numpy as np
import pandas as pd

import catboost as ctb

from sklearn.model_selection import train_test_split
from sklearn.metrics import recall_score, precision_score, roc_auc_score, accuracy_score, f1_score
```

Классы и функции для задания

In [1]:
```
def evaluate_model(model, X_train, y_train, X_test, y_test):
    """
    Обучить и оценить модель.
    """
    model = ctb.CatBoostClassifier(cat_features=cat_feats)
    model.fit(X_train, y_train, verbose=False)
    y_pred = model.predict(X_test)

    f1 = f1_score(y_test, y_pred)
    roc = roc_auc_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred, average='binary')
    rec = recall_score(y_test, y_pred, average='binary')

    return {'f1': [f1], 'roc-auc': [roc], 'precision': [prec], 'recall': [rec]}
```

Задание 1

Взять любой набор данных для бинарной классификации (можно скачать один из модельных с https://archive.ics.uci.edu/ml/datasets.php).

Решение Задания 1

Kaggle Dataset - HR Analytics: Job Change of Data Scientists:

- https://www.kaggle.com/arashnic/hr-analytics-job-change-of-data-scientists

In [3]:
```
df = pd.read_csv("data/aug_train.csv")
df.head()
```

Out[3]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_s
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	>20	N
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM	15	50-
2	11561	city_21	0.624	NaN	No relevent experience	Full time course	Graduate	STEM	5	N
3	33241	city_115	0.789	NaN	No relevent experience	NaN	Graduate	Business Degree	<1	N
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment	Masters	STEM	>20	50-

In [4]:
```
df.shape
```

Out[4]:
```
(19158, 14)
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19158 entries, 0 to 19157
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   enrollee_id             19158 non-null  int64
 1   city                    19158 non-null  object
 2   city_development_index  19158 non-null  float64
 3   gender                  14650 non-null  object
 4   relevent_experience     19158 non-null  object
 5   enrolled_university     18772 non-null  object
 6   education_level         18698 non-null  object
 7   major_discipline        16345 non-null  object
 8   experience              19093 non-null  object
```

```
 9   company_size            13220 non-null  object
 10  company_type            13018 non-null  object
 11  last_new_job            18735 non-null  object
 12  training_hours          19158 non-null  int64
 13  target                  19158 non-null  float64
dtypes: float64(2), int64(2), object(10)
memory usage: 2.0+ MB
```

Ненужный признак

In [6]:
```
df = df.drop(columns=['enrollee_id'])
```

Конвертируем таргет в int

In [7]:
```
df['target'] = df['target'].astype(int)
```

Смотрим баланс таргета

In [8]:
```
df['target'].value_counts()
```

Out[8]:
```
0    14381
1     4777
Name: target, dtype: int64
```

Задание 2

Сделать feature engineering.

Решение Задания 2

Просто заменяем все на самое частое значение (моду), т.к. все признаки с пропущенными значениями категориальные.

In [9]:
```
for col in df.select_dtypes('object').columns:
    df[col] = df[col].fillna(df[col].value_counts().index[0])
```

Делим данные на трейн и тест.

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(df.drop(columns=['target']), df['target'], test_size=0.2, random_state=42)
```

Задание 3

Обучить любой классификатор (какой вам нравится).

Решение Задания 3

In [11]:
```
cat_feats = ['city', 'gender', 'relevent_experience', 'enrolled_university', 'education_level',
             'major_discipline', 'experience', 'company_size', 'company_type', 'last_new_job']
```

In [13]:
```
metrics = pd.DataFrame(evaluate_model(ctb.CatBoostClassifier(cat_features=cat_feats),
                                      X_train,
                                      y_train,
                                      X_test,
                                      y_test))
```

In [14]:
```
metrics
```

Out[14]:

	f1	roc-auc	precision	recall
0	0.504113	0.669586	0.572	0.45063

Задание 4

Далее разделить ваш набор данных на два множества: P (positives) и U (unlabeled). Причем брать нужно не все положительные (класс 1) примеры, а только лишь часть.

Решение Задания 4

In [15]:
```
def create_unlabeled(df, pos_frac=0.2):
    """
    Сэмплирует долю pos_frac наблюдений класса 1 как positive, остальные как unlabeled.
    """
    sdf = df.copy()
    pos_mask = (df['target'] == 1)
    pos_ind = df[pos_mask].sample(frac=pos_frac).index
    unlab_ind = df[~df.index.isin(pos_ind)].index

    # Помечаем данные признаком is_labeled - Positive = 1, Unlabeled = 0
    df.loc[pos_ind, 'is_labeled'] = 1
```

```
    df.loc[unlab_ind, 'is_labeled'] = 0
    df['is_labeled'] = df['is_labeled'].astype(int)
    return df
```

Возьмем 20% наблюдений положительного класса как positive, остальные возьмем как unlabeled.

In [17]:
```
rns_df = create_unlabeled(df, pos_frac=0.2)
```

In [18]:
```
rns_df.head(3)
```

Out[18]:

	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	compan
0	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	>20	50-99	I
1	city_40	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM	15	50-99	I
2	city_21	0.624	Male	No relevent experience	Full time course	Graduate	STEM	5	50-99	I

Задание 5

Применить random negative sampling для построения классификатора в новых условиях.

Решение Задания 5

In [19]:
```
def get_rns_samples(rns_df):
    """
    Создает тренировочную и тестовую выборки для RNS на основе признака is_labeled.
    """
    rns_df = rns_df.sample(frac=1)

    pos_sample = rns_df[rns_df['is_labeled'] == 1]
    neg_sample = rns_df[rns_df['is_labeled'] == 0][:pos_sample.shape[0]]
    train_samples = pd.concat([neg_sample, pos_sample]).sample(frac=1)
    test_samples = rns_df[rns_df['is_labeled'] == 0][pos_sample.shape[0]:]

    return train_samples, test_samples
```

In [20]:
```
train_samples, test_samples = get_rns_samples(rns_df)
```

In [21]:
```
metrics_task5 = evaluate_model(ctb.CatBoostClassifier(cat_features=cat_feats),
                               train_samples.iloc[:, :-2],
                               train_samples['is_labeled'],
                               test_samples.iloc[:, :-2],
                               test_samples['target'])
```

In [22]:
```
metrics = metrics.append(pd.DataFrame(metrics_task5))
```

Задание 6

Сравнить качество с решением из пункта 4 (построить отчет - таблицу метрик).

Решение Задания 6

In [23]:
```
metrics.index = ['normal', 'RNS']
```

In [24]:
```
metrics
```

Out[24]:

	f1	roc-auc	precision	recall
normal	0.504113	0.669586	0.572000	0.450630
RNS	0.536716	0.728180	0.438287	0.692159

RNS справился даже немного лучше, чем обычная модель. Интересно то, что повысился recall и понизилась precision - модели стало сложнее различать между классами, поэтому она начала относить больше наблюдений к положительному классу.

Задание 7

Поэкспериментировать с долей P на шаге 5 (как будет меняться качество модели при уменьшении/увеличении размера P).

Решение Задания 7

In [26]:
```
rns_metrics = pd.DataFrame(columns=['f1', 'roc-auc', 'precision', 'recall'])

fracs = np.linspace(0.1, 0.9, 9)
for frac in fracs:
    train_samples, test_samples = get_rns_samples(create_unlabeled(df, pos_frac=frac))
    frac_metrics = evaluate_model(ctb.CatBoostClassifier(cat_features=cat_feats),
                                  train_samples.iloc[:, :-2],
                                  train_samples['is_labeled'],
                                  test_samples.iloc[:, :-2],
                                  test_samples['target'])
    rns_metrics = rns_metrics.append(pd.DataFrame(frac_metrics))
```

In [27]:
```
rns_metrics.index = fracs
```

In [28]:
```
rns_metrics
```

Out[28]:

	f1	roc-auc	precision	recall
0.1	0.540461	0.721235	0.424727	0.742891
0.2	0.532138	0.731384	0.416212	0.737569
0.3	0.521309	0.739109	0.408261	0.720938
0.4	0.486726	0.733078	0.371090	0.707051
0.5	0.444347	0.735743	0.318315	0.735593
0.6	0.403962	0.740023	0.279491	0.728316
0.7	0.333607	0.738729	0.215536	0.737750
0.8	0.259835	0.736042	0.157724	0.736915
0.9	0.150490	0.730634	0.084099	0.714715

По ROC-AUC, precision и recall лучшая доля сэмплинга - 0.1, по F1 score - 0.6.