GeekBrains, ML in Business

Lesson 5 Homework

Ссылки:

- http://hyperopt.github.io/hyperopt/
- https://arxiv.org/pdf/1907.03947.pdf
- https://arxiv.org/pdf/1802.02301.pdf
- https://arxiv.org/list/stat.ML/recent
- https://scikit-learn.org/stable/modules/grid_search.html
- https://scikit-learn.org/stable/modules/compose.html

Импорт библиотек

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

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import fbeta_score, precision_recall_curve, confusion_matrix
```

Классы и функции для задания

In [2]:
```
class FeatureSelector(BaseEstimator, TransformerMixin):
    def __init__(self, column):
        self.column = column

    def fit(self, X, y=None):
        return self

    def transform(self, X, y=None):
        return X[self.column]

class NumberSelector(BaseEstimator, TransformerMixin):
    """
    Transformer to select a single column from the data frame to perform additional transformations on
    Use on numeric columns in the data
    """
    def __init__(self, key):
        self.key = key

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        return X[[self.key]]

class OHEEncoder(BaseEstimator, TransformerMixin):
    def __init__(self, key):
        self.key = key
        self.columns = []

    def fit(self, X, y=None):
        self.columns = [col for col in pd.get_dummies(X, prefix=self.key).columns]
        return self

    def transform(self, X):
        X = pd.get_dummies(X, prefix=self.key)
        test_columns = [col for col in X.columns]
        for col_ in self.columns:
            if col_ not in test_columns:
                X[col_] = 0
        return X[self.columns]
```

In [3]:
```
def threshold_by_fbeta(y_test: pd.Series, y_pred: list, *, beta: int = 1) -> tuple:
    precision, recall, thresholds = precision_recall_curve(y_test, y_pred)
    fbeta = ((1 + beta**2) * precision * recall) / (beta**2 * precision + recall)
    index = np.argmax(fbeta)
    return thresholds[index], fbeta[index]
```

Чтение данных

In [4]:
```
df = pd.read_csv("churn_data.csv")
df.head(3)
```

Out[4]:

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Estimatec

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Estimatec
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113

Делим данные на трейн и тест.

In [5]:
```
X_train, X_test, y_train, y_test = train_test_split(df, df['Exited'], random_state=0)
```

Задание 1, 2

1. Для пайплайна Case1 поэкспериментировать с разными моделями: 1 - бустинг, 2 - логистическая регрессия (не забудьте здесь добавить в cont_transformer стандартизацию - нормирование вещественных признаков).
2. Отобрать лучшую модель по метрикам (кстати, какая по вашему мнению здесь наиболее подходящая DS-метрика).

Решение Заданий 1, 2

В задаче оттока нужная метрика во многом зависит от стоимости и прибыли удержания старых клиентов, получения новых клиентов. Предположим, что удерживать старых клиентов прибыльнее, чем получать новых. Иначе нет смысла решать задачу оттока. Также предположим, что каждый TP приносит 2 доллара, а каждый FP отнимает 1 доллар (как в кейсе с вебинара).

В таком случае есть смысл использовать F_β-score со значением $\beta = 2$ - ошибка FN в 2 раза хуже ошибки FP. Таким образом нам экономически выгоднее будет найти больше оттекающих клиентов, даже если некоторые предсказания будут ошибочны.

Обучим модели случайного леса (изначальная), градиентного бустинга и логистической регрессии. При этом, добавим в пайплайн нормирование признаков классом MinMaxScaler для всех моделей, чтобы датасет был одинаковый для всех моделей.

Признаки по видам.

In [6]:
```
categorical_columns = ['Geography', 'Gender', 'Tenure', 'HasCrCard', 'IsActiveMember']
continuous_columns = ['CreditScore', 'Age', 'Balance', 'NumOfProducts', 'EstimatedSalary']
```

Обработка категориальных и числовых признаков.

In [7]:
```
final_transformers = []

for cat_col in categorical_columns:
    cat_transformer = Pipeline([
                ('selector', FeatureSelector(column=cat_col)),
                ('ohe', OHEEncoder(key=cat_col))
            ])
    final_transformers.append((cat_col, cat_transformer))

for cont_col in continuous_columns:
    cont_transformer = Pipeline([
                ('selector', NumberSelector(key=cont_col)),
                ('scaler', MinMaxScaler())
            ])
    final_transformers.append((cont_col, cont_transformer))
```

In [8]:
```
feats = FeatureUnion(final_transformers)
```

Модели.

In [9]:
```
classifiers = [
    RandomForestClassifier(random_state=42),
    GradientBoostingClassifier(random_state=42),
    LogisticRegression(random_state=42)
]
```

Собираем пайплайны и обучаем их.

In [10]:
```
pipelines = []
metrics = {'threshold': [], 'fbeta': []}
```

In [11]:
```
for clf in classifiers:
    pipeline = Pipeline([
        ('features', feats),
        ('classifier', clf),
    ])
    pipeline.fit(X_train, y_train)

    y_pred = pipeline.predict_proba(X_test)[:, 1]
    threshold, fbeta_score = threshold_by_fbeta(y_test, y_pred, beta=2)

    metrics['threshold'].append(threshold)
```

```
    metrics['fbeta'].append(fbeta_score)
    pipelines.append(pipeline)
```

Смотрим метрики.

In [12]:
```
metrics_df = pd.DataFrame(metrics, index=[clf.__class__.__name__ for clf in classifiers])
metrics_df.sort_values('fbeta', ascending=False)
```

Out[12]:

	threshold	fbeta
GradientBoostingClassifier	0.154589	0.717822
RandomForestClassifier	0.150000	0.693304
LogisticRegression	0.135475	0.636624

Лучшей моделью оказался градиентный бустинг (индекс 1 в списке pipelines).

In [13]:
```
best_model = pipelines[1]
```

Задание 3

Для отобранной модели (на отложенной выборке) сделать оценку экономической эффективности при тех же вводных, как в вопросе 2 (1 доллар на привлечение, 2 доллара - с каждого правильно классифицированного (True Positive) удержанного). (подсказка) нужно посчитать FP/TP/FN/TN для выбранного оптимального порога вероятности и посчитать выручку и траты.

Решение Задания 3

Прибыль TP = 2, FP = -1, TN = 0, FN = 0 (т.к. для TP у нас стоит прибыль).

In [14]:
```
y_pred_best = best_model.predict_proba(X_test)[:, 1]
_, _, thresholds = precision_recall_curve(y_test, y_pred)
```

In [15]:
```
def economic_metrics(y_pred_probas: list, thresholds: list) -> pd.DataFrame:
    profits = []
    expenses = []
    for thr in thresholds:
        preds = (y_pred_best >= thr).astype(bool)
        conf = confusion_matrix(y_test, preds)
        TP = conf[1][1]
        FP = conf[0][1]
        profits.append(TP * 2)
        expenses.append(FP * -1)

    results_df = pd.DataFrame({'threshold': thresholds, 'profit': profits, 'expense': expenses})
    results_df['total'] = results_df['profit'] + results_df['expense']
    return results_df
```

In [16]:
```
results = economic_metrics(y_pred_best, thresholds)
```

In [17]:
```
results.sort_values('total', ascending=False)
```

Out[17]:

	threshold	profit	expense	total
2178	0.407250	608	-128	480
2175	0.406141	608	-128	480
2179	0.407268	608	-128	480
2180	0.407311	608	-128	480
2177	0.406888	608	-128	480
...	...	...	...	...
4	0.014457	1014	-1960	-946
3	0.014286	1014	-1963	-949
2	0.013231	1016	-1983	-967
1	0.011778	1018	-1989	-971
0	0.010434	1018	-1990	-972

2500 rows × 4 columns

Одним из лучших пороговых значений оказалось значение 0.40725 с итоговой прибылью 480 долларов.

Задание 4*

Провести подбор гиперпараметров лучшей модели по итогам 2-3.

Решение Задания 4*

Параметры.

In [18]:
```
params = {
    'classifier__n_estimators': [100, 200, 300],
    'classifier__min_samples_leaf': [1, 2, 3],
    'classifier__max_depth': [3, 4, 5]
}
```

In [19]:
```
grid = GridSearchCV(best_model,
                    param_grid=params,
                    cv=5,
                    refit=False)
```

Запускаем перебор.

In [20]:
```
search = grid.fit(X_train, y_train)
search.best_params_
```

Out[20]:
```
{'classifier__max_depth': 5,
 'classifier__min_samples_leaf': 3,
 'classifier__n_estimators': 100}
```

Обучаем оптимальную модель.

In [21]:
```
optimal_model = Pipeline([
    ('features', feats),
    ('classifier', GradientBoostingClassifier(n_estimators=100, max_depth=5, min_samples_leaf=3, random_state=42)),
])
```

In [22]:
```
optimal_model.fit(X_train, y_train)
```

Out[22]:
```
Pipeline(steps=[('features',
                 FeatureUnion(transformer_list=[('Geography',
                                                 Pipeline(steps=[('selector',
                                                                  FeatureSelector(column='Geography')),
                                                                 ('ohe',
                                                                  OHEEncoder(key='Geography'))])),
                                                ('Gender',
                                                 Pipeline(steps=[('selector',
                                                                  FeatureSelector(column='Gender')),
                                                                 ('ohe',
                                                                  OHEEncoder(key='Gender'))])),
                                                ('Tenure',
                                                 Pipeline(steps=[('selector',
                                                                  FeatureSelector(column='Tenu...
                                                                  NumberSelector(key='Balance')),
                                                                 ('scaler',
                                                                  MinMaxScaler())])),
                                                ('NumOfProducts',
                                                 Pipeline(steps=[('selector',
                                                                  NumberSelector(key='NumOfProducts')),
                                                                 ('scaler',
                                                                  MinMaxScaler())])),
                                                ('EstimatedSalary',
                                                 Pipeline(steps=[('selector',
                                                                  NumberSelector(key='EstimatedSalary')),
                                                                 ('scaler',
                                                                  MinMaxScaler())]))])),
                ('classifier',
                 GradientBoostingClassifier(max_depth=5, min_samples_leaf=3,
                                            random_state=42))])
```

Предсказываем и считаем F-Beta метрику.

In [23]:
```
y_pred_optim = optimal_model.predict_proba(X_test)[:, 1]
```

In [24]:
```
threshold, fbeta_score = threshold_by_fbeta(y_test, y_pred_optim, beta=2)
```

In [25]:
```
threshold, fbeta_score
```

Out[25]:
```
(0.12823119166694474, 0.7170418006430869)
```

F-Beta score получился немного хуже, чем у изначальной модели.

Задание 5*

Еще раз провести оценку экономической эффективности.

Решение Задания 5*

In [26]:
```
_, _, thresholds_optim = precision_recall_curve(y_test, y_pred_optim)
```

In [27]:
```
results_optim = economic_metrics(y_pred_best, thresholds_optim)
```

In [28]:
```
results_optim.sort_values('total', ascending=False)
```

Out[28]:

	threshold	profit	expense	total
2020	0.404927	608	-128	480
2021	0.404966	608	-128	480
2022	0.406520	608	-128	480
2023	0.407940	608	-128	480
2024	0.409099	606	-127	479
...	...	...	...	...
1	0.009232	1018	-1991	-973
6	0.010006	1018	-1991	-973
7	0.010161	1018	-1991	-973
8	0.010165	1018	-1991	-973
0	0.008998	1018	-1991	-973

2482 rows × 4 columns

Одним из лучших значений является значение 0.40652 со все такой же итоговой прибылью 480 долларов.