lesson 5

1. Для нашего 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)
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.
4. (опционально) Провести подбор гиперпараметров лучшей модели по итогам 2-3
5. (опционально) Еще раз провести оценку экономической эффективности

Case 1

Ввод [1]:
```
import pandas as pd
import numpy as np
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
#from sklearn.feature_extraction.text import TfidfVectorizer
import itertools
from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

from sklearn.pipeline import Pipeline
from sklearn.pipeline import FeatureUnion

from sklearn.metrics import f1_score, roc_auc_score, precision_score
from sklearn.metrics import classification_report, precision_recall_curve, confusion_matrix

%matplotlib inline
```

Ввод [2]:
```
df = pd.read_csv("churn_data.csv")
df.head(3)
```

Out[2]:

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

Ввод [3]:
```
#разделим данные на train/test
X_train, X_test, y_train, y_test = train_test_split(df, df['Exited'], random_state=0)
```

Ввод [4]:
```
#соберем наш простой pipeline, но нам понадобится написать класс для выбора нужного поля
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]
```

Ввод [5]:
```
categorical_columns = ['Geography', 'Gender', 'Tenure', 'HasCrCard', 'IsActiveMember']
continuous_columns = ['CreditScore', 'Age', 'Balance', 'NumOfProducts', 'EstimatedSalary']
```

Ввод [6]:
```
final_transformers = list()

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)),
                ('std_scaler', StandardScaler())
            ])
    final_transformers.append((cont_col, cont_transformer))
```

Ввод [7]:
```
feats = FeatureUnion(final_transformers)

feature_processing = Pipeline([('feats', feats)])
```

Ввод [8]:
```
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression


pipeline_rfc = Pipeline([
    ('features',feats),
    ('classifier', RandomForestClassifier(random_state=42))
])

pipeline_gbc = Pipeline([
    ('features',feats),
    ('classifier', GradientBoostingClassifier(random_state=42))
])



pipeline_lr = Pipeline([
    ('features',feats),
    ('classifier', LogisticRegression(random_state = 42)),
])

pipelines = [pipeline_rfc, pipeline_gbc, pipeline_lr]
```

Ввод [9]:
```
for pipeline in pipelines:

    #обучим наш пайплайн
    pipeline.fit(X_train, y_train)

    #наши прогнозы для тестовой выборки
    preds = pipeline.predict_proba(X_test)[:, 1]

    precision, recall, thresholds = precision_recall_curve(y_test, preds)

    beta = 1
    fscore = ((1 + beta ** 2) * precision * recall) / ((beta ** 2) * precision + recall)
    # locate the index of the largest f score
    ix = np.argmax(fscore)
    print(pipeline.steps[1][1])
    print('Best Threshold=%f, F-Score=%.3f, Precision=%.3f, Recall=%.3f' % (thresholds[ix],
```

```
                                                                    fscore[ix],
                                                                    precision[ix],
                                                                    recall[ix]))
    print()
```

```
RandomForestClassifier(random_state=42)
Best Threshold=0.380000, F-Score=0.641, Precision=0.654, Recall=0.629

GradientBoostingClassifier(random_state=42)
Best Threshold=0.408508, F-Score=0.646, Precision=0.704, Recall=0.597

LogisticRegression(random_state=42)
Best Threshold=0.289522, F-Score=0.510, Precision=0.462, Recall=0.568
```

Problem 2

- Лучшая модель GradientBoostingClassifier. Выбор делал путем сравнения метрик F1-score.
- Думаю, что нужно максимизировать F-score с другим коэффициетом бета (отличным от 1). В какую сторону менять бета подскажет расчет экономической эффективности.

Problem 3

Ввод [10]:
```
# функция, основанная на результате ответа на вопрос 2
def profit(cnf_matrix, a=1, b=2):
    result = 2 * b * cnf_matrix[1, 1]  - a * (cnf_matrix[0, 1] + cnf_matrix[1, 1])
    return result
```

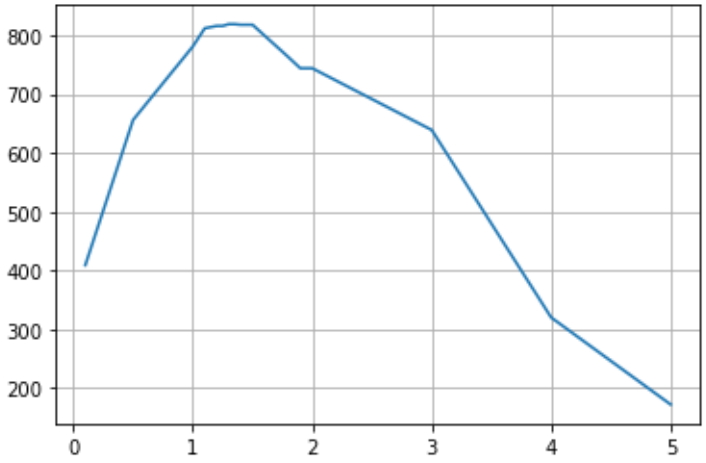
Ввод [11]:
```
preds = pipeline_gbc.predict_proba(X_test)[:, 1]
betas = [0.1, 0.5, 1, 1.1, 1.2, 1.25, 1.3, 1.35, 1.4, 1.5, 1.8, 1.9, 2,3, 4, 5]
profits = []
max_profit = 0
beta_m = 0

for beta in betas:
    precision, recall, thresholds = precision_recall_curve(y_test, preds)
    fscore = ((1 + beta ** 2) * precision * recall) / ((beta ** 2) * precision + recall)
```

```
    ix = np.argmax(fscore)
    cnf_matrix = confusion_matrix(y_test, preds>thresholds[ix])
    current_profit = profit(cnf_matrix)
    profits.append(current_profit)
    if current_profit > max_profit:
        max_profit = current_profit
        beta_m = beta


print(f'beta={beta_m}, max_profit={max_profit}')
plt.plot(betas, profits)
plt.grid()
```

```
beta=1.3, max_profit=819
```



Максимальный экономический эффект проявляется при beta = 1.3.

Ввод [12]:
```
beta = 1.3
precision, recall, thresholds = precision_recall_curve(y_test, preds)
fscore = ((1 + beta ** 2) * precision * recall) / ((beta ** 2) * precision + recall)
ix = np.argmax(fscore)
cnf_matrix = confusion_matrix(y_test, preds>thresholds[ix])
```

Вопрос 1: объясните своими словами смысл метрик Precison, Recall *

1. Какова их взаимосвязь и как с ними связан порог вероятности?
2. Можно ли подобрать порог так, что recall будет равен 1? Что при этом будет с precision
3. Аналогичный вопрос про precision

Ответ:

- Precison - точность. Показывает долю правильных ответов классификатора для объектов, отнесенных к классу1.
- Recall - полнота. Показывает долю правильно отнесенных объектов к классу 1 от всех действительных объектов класса 1.

1. При увеличении Recall уменьшается Precision и наоборот. Порог вероятности - это планка для отнесения объекта к классу 1. Большой порог вероятности дает большой Precision и маленький Recall, а маленький - маленький Precision и большой Recall.
2. Можно подобрать порог, при котором Recall будет равен 1 (при некотором малом значении порога вероятности - ближе к 0). При данном условии Precision будет иметь значение близкое к нулю.
3. Можно подобрать порог, при котором Precision будет равен 1 (при некотором большом значении порога вероятности - ближе к 1). При данном условии Recall будет иметь значение близкое к нулю.

Вопрос 2: предположим, что на удержание одного пользователя у нас уйдет 1 доллар. При этом средняя ожидаемая прибыль с каждого TP (true positive) - 2 доллара. Оцените качество модели выше с учетом этих данных и ответьте на вопрос, является ли она потенциально экономически целесообразной?

Ответ:

Ввод [13]:
```
a = 1 # стоимость удержания
b = 2 # средняя ожидаемая прибыль с каждого TP
```

Потери без попыток удержания:

Ввод [14]:
```
loss_non_churn = b * (cnf_matrix[1, 0] + cnf_matrix[1, 1])
```

Потери удержания с учетом попыток удержания:

Ввод [15]:
```
loss_churn = b * cnf_matrix[1, 0] + a * (cnf_matrix[0, 1] + cnf_matrix[1, 1]) - b * cnf_matrix[1, 1]
```

Экономический эффект от попыток удержания

Ввод [16]:
```
profit_all = -(loss_churn - loss_non_churn)
profit_all
```

Out[16]: 819

Модель потенциально экономически целесообразна.

Ввод [ ]:

Problems 4-5

Ввод [17]:
```
params_gbc = {'classifier__loss': ['exponential'],
              'classifier__learning_rate': [0.1, 0.05],
              'classifier__n_estimators': [400, 300],
              'classifier__subsample': [0.7, 0.8],
              'classifier__min_samples_leaf': [9, 10, 11],
              'classifier__max_leaf_nodes': [6, 7, 8]}
```

Ввод [18]:
```
from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(pipeline_gbc,
                    param_grid=params_gbc,
                    cv=6,
                    refit=False)

#search = grid.fit(X_train, y_train)
#search.best_params_
```

Ввод [19]:
```
pipeline_gbc = Pipeline([
    ('features',feats),
    ('classifier', GradientBoostingClassifier(random_state=42,
                                               learning_rate=0.05,
                                               loss='exponential',
```

```
                                                max_leaf_nodes=6,
                                                min_samples_leaf=10,
                                                n_estimators=300,
                                                subsample=0.7))
])

pipeline_gbc.fit(X_train, y_train)
```

Out[19]:
```
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='NumOfProducts')),
                                                                 ('std_scaler',
                                                                  StandardScaler())])),
                                                ('EstimatedSalary',
                                                 Pipeline(steps=[('selector',
                                                                  NumberSelector(key='EstimatedSalary')),
                                                                 ('std_scaler',
                                                                  StandardScaler())]))])),
                ('classifier',
                 GradientBoostingClassifier(learning_rate=0.05,
                                            loss='exponential',
                                            max_leaf_nodes=6,
                                            min_samples_leaf=10,
                                            n_estimators=300, random_state=42,
                                            subsample=0.7))])
```

Ввод [21]:
```
preds = pipeline_gbc.predict_proba(X_test)[:, 1]
betas = [0.1, 0.5, 1, 1.1, 1.2, 1.25, 1.3, 1.35, 1.4, 1.5, 1.8, 1.9, 2,3, 4, 5]
profits = []
```

```
max_profit = 0
beta_m = None

for beta in betas:
    precision, recall, thresholds = precision_recall_curve(y_test, preds)
    fscore = ((1 + beta ** 2) * precision * recall) / ((beta ** 2) * precision + recall)
    ix = np.argmax(fscore)
    cnf_matrix = confusion_matrix(y_test, preds>thresholds[ix])
    current_profit = profit(cnf_matrix)
    profits.append(current_profit)
    if current_profit > max_profit:
        max_profit = current_profit
        beta_m = beta


print(f'beta={beta_m}, max_profit={max_profit}')
plt.plot(betas, profits)
plt.grid()
```

```
beta=1.25, max_profit=821
```