Lesson 6

1. взять любой набор данных для бинарной классификации (можно скачать один из модельных с [https://archive.ics.uci.edu/ml/datasets.php (https://archive.ics.uci.edu/ml/datasets.php)](https://archive.ics.uci.edu/ml/datasets.php))
2. сделать feature engineering
3. обучить любой классификатор (какой вам нравится)
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
5. применить random negative sampling для построения классификатора в новых условиях
6. сравнить качество с решением из пункта 4 (построить отчет - таблицу метрик)
7. поэкспериментировать с долей P на шаге 5 (как будет меняться качество модели при уменьшении/увеличении размера P)

Ввод [ ]:

```
```

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
from sklearn.model_selection import GridSearchCV
%matplotlib inline
```

Ввод [2]:

```
data = pd.read_csv('airline_passenger_satisfaction.csv', index_col=False)
```

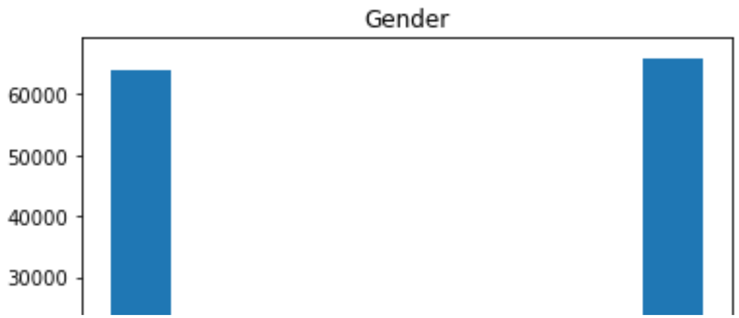
Ввод [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129880 entries, 0 to 129879
Data columns (total 24 columns):
 #   Column                            Non-Null Count   Dtype 
---  ------                            --------------   ----- 
 0   Unnamed: 0                        129880 non-null  int64 
 1   Gender                            129880 non-null  object
 2   customer_type                     129880 non-null  object
 3   age                               129880 non-null  int64 
 4   type_of_travel                    129880 non-null  object
 5   customer_class                    129880 non-null  object
 6   flight_distance                   129880 non-null  int64 
 7   inflight_wifi_service             129880 non-null  int64 
 8   departure_arrival_time_convenient 129880 non-null  int64 
 9   ease_of_online_booking            129880 non-null  int64 
 10  gate_location                     129880 non-null  int64 
 11  food_and_drink                    129880 non-null  int64 
 12  online_boarding                   129880 non-null  int64 
 13  seat_comfort                      129880 non-null  int64 
 14  inflight_entertainment            129880 non-null  int64 
```

Ввод [4]:
```
data['satisfaction'].replace({'satisfied': 1, 'neutral or dissatisfied': 0}, inplace=True)
y = data['satisfaction']
X = data.drop(['satisfaction', 'Unnamed: 0'], axis=1)
```

Ввод [5]:
```
for col in X.columns:
    print(data[col].unique())
    plt.hist(data[col])
    plt.title(col)
    plt.show()
```

```
['Male' 'Female']
```



Ввод [6]:
```
#разделим данные на train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Ввод [7]:
```
#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):
```

```python
        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 [8]:

```python
categorical_columns = ['Gender', 'customer_type', 'type_of_travel', 'customer_class']
continuous_columns = ['age', 'flight_distance', 'inflight_wifi_service', 'departure_arrival_time_convenient',
                      'ease_of_online_booking', 'gate_location', 'food_and_drink', 'online_boarding',
                      'seat_comfort', 'inflight_entertainment', 'onboard_service', 'leg_room_service',
                      'baggage_handling', 'checkin_service', 'inflight_service', 'cleanliness',
                      'departure_delay_in_minutes', 'arrival_delay_in_minutes']
```

Ввод [9]:

```python
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))
            ])
    final_transformers.append((cont_col, cont_transformer))
```

Ввод [10]:

```python
feats = FeatureUnion(final_transformers)
feature_processing = Pipeline([('feats', feats)])
```

Ввод [11]:

```python
from sklearn.ensemble import GradientBoostingClassifier
```

```python
from catboost import CatBoostClassifier

pipeline = Pipeline([
    ('features',feats),
    ('classifier', CatBoostClassifier(random_state=42))
])
```

Ввод [12]:

```python
#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]))

result = {'look-alike': ['False',], 'P_fraction': [None,], 'F1-Score': [fscore[ix],],
          'Precision': [precision[ix],], 'Recall': [recall[ix],]}
```

```
Learning rate set to 0.070676
0:	learn: 0.5989156	total: 203ms	remaining: 3m 22s
1:	learn: 0.5032592	total: 247ms	remaining: 2m 3s
2:	learn: 0.4276257	total: 280ms	remaining: 1m 33s
3:	learn: 0.3829203	total: 343ms	remaining: 1m 25s
```

Ввод [ ]:

Ввод [13]:

```
data_train, data_test = train_test_split(data, test_size=0.3)
y_test = data_test['satisfaction']
X_test = data_test.drop('satisfaction', axis=1)

pos_ind = data_train['satisfaction'] == 1
P_start = data_train.loc[pos_ind]
U_start = data_train.loc[~pos_ind]
```

Ввод [14]:

```
for P_fraction in np.arange(0.1, 1, 0.1):
    pipeline = Pipeline([
                    ('features',feats),
                    ('classifier', CatBoostClassifier(random_state=42))
                    ])
    P, P_u = train_test_split(P_start, train_size=P_fraction, random_state=42)
    U = pd.concat([P_u, U_start], axis=0).sample(frac=1)
    U['satisfaction'] = 0
    data_train_new = pd.concat([U, P], axis=0)
    y_train = data_train_new['satisfaction']
    X_train = data_train_new.drop('satisfaction', axis=1)

    #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]))

    result['look-alike'].append('True')
    result['F1-Score'].append(fscore[ix])
    result['Precision'].append(precision[ix])
    result['Recall'].append(recall[ix])
    result['P_fraction'].append(P_fraction)
```

```
Learning rate set to 0.070676
0:      learn: 0.5935732        total: 42.4ms   remaining: 42.3s
1:      learn: 0.5149324        total: 69.8ms   remaining: 34.8s
2:      learn: 0.4428932        total: 118ms    remaining: 39.2s
3:      learn: 0.3919628        total: 181ms    remaining: 45.1s
4:      learn: 0.3513210        total: 214ms    remaining: 42.5s
5:      learn: 0.3154363        total: 243ms    remaining: 40.2s
6:      learn: 0.2855535        total: 269ms    remaining: 38.1s
7:      learn: 0.2626538        total: 303ms    remaining: 37.6s
8:      learn: 0.2468072        total: 325ms    remaining: 35.8s
9:      learn: 0.2334681        total: 376ms    remaining: 37.2s
10:     learn: 0.2220646        total: 416ms    remaining: 37.4s
11:     learn: 0.2120329        total: 473ms    remaining: 38.9s
12:     learn: 0.2042634        total: 508ms    remaining: 38.6s
13:     learn: 0.1967121        total: 550ms    remaining: 38.7s
14:     learn: 0.1920680        total: 573ms    remaining: 37.6s
15:     learn: 0.1867125        total: 611ms    remaining: 37.6s
16:     learn: 0.1822703        total: 641ms    remaining: 37.1s
17:     learn: 0.1775178        total: 674ms    remaining: 36.8s
```

Ввод [15]:

```
pd.DataFrame(result)
```

Out[15]:

	look-alike	P_fraction	F1-Score	Precision	Recall
0	False	NaN	0.958670	0.974169	0.943656
1	True	0.1	0.919120	0.927386	0.911000
2	True	0.2	NaN	0.000000	0.000000
3	True	0.3	0.941545	0.952404	0.930930
4	True	0.4	0.945045	0.956829	0.933548
5	True	0.5	0.948522	0.962972	0.934499

	look-alike	P_fraction	F1-Score	Precision	Recall
6	True	0.6	0.950481	0.967466	0.934083
7	True	0.7	0.952620	0.967313	0.938366
8	True	0.8	0.954185	0.972334	0.936701

Ввод [ ]: