Рекомендация тарифов

В вашем распоряжении данные о поведении клиентов, которые уже перешли на эти тарифы (из проекта курса «Статистический анализ данных»). Нужно построить модель для задачи классификации, которая выберет подходящий тариф. Предобработка данных не понадобится — вы её уже сделали.

Постройте модель с максимально большим значением *accuracy*. Чтобы сдать проект успешно, нужно довести долю правильных ответов по крайней мере до 0.75. Проверьте *accuracy* на тестовой выборке самостоятельно.

Цель исследования — найти модель с максимально большим значением *accuracy* (не меньше 0.75)

Данные о поведении клиентов получим из файла users_behavior.csv. Предобработка данных уже проведена ранее

Исследование пройдёт в пять этапов:

- Обзор и изучение данных
- Разделение данных на выборки
- Исследование различных моделей и подбор лучших гиперпараметров
  - Decision Tree (Дерево решений)
  - Random Forest (Случайный лес)
  - Logistic Regression (Логистическая регрессия)
- Проверка итоговой модели на тестовой выборке
- Проверка модели на адекватность

Откроем и изучим файл

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.dummy import DummyClassifier

data = pd.read_csv('datasets/users_behavior.csv')
r_state = 12345
```

In [2]:
```
data.info()
display(data.sample(5, random_state=r_state))

print('Количество пропусков по столбцам:')
print()

for col in data.columns:
    nmv = data[col].isna().sum()
    pmv = nmv/len(data)
    print('{} - {} шт. - {}%'.format(col, nmv, round(pmv*100, 2)))

print()
print('Количество явных дубликатов:', data.duplicated().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3214 entries, 0 to 3213
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   calls     3214 non-null   float64
 1   minutes   3214 non-null   float64
 2   messages  3214 non-null   float64
 3   mb_used   3214 non-null   float64
 4   is_ultra  3214 non-null   int64
dtypes: float64(4), int64(1)
memory usage: 125.7 KB
```

	calls	minutes	messages	mb_used	is_ultra
1415	82.0	507.89	88.0	17543.37	1
916	50.0	375.91	35.0	12388.40	0
1670	83.0	540.49	41.0	9127.74	0
686	79.0	562.99	19.0	25508.19	1
2951	78.0	531.29	20.0	9217.25	0

Количество пропусков по столбцам:

calls - 0 шт. - 0.0%
minutes - 0 шт. - 0.0%
messages - 0 шт. - 0.0%
mb_used - 0 шт. - 0.0%
is_ultra - 0 шт. - 0.0%

Количество явных дубликатов: 0

- Предобработка данных выполнена ранее
- Данные состоят из 3214 объектов
- Пропущенных значений нет
- Явные дубликаты отсутствуют
- Данные имеют 5 признаков (1 целевой и 4 вспомогательных)

Разобьем данные на выборки

Определим целевой и вспомогательные признаки

In [3]:
```
features = data.drop(['is_ultra'], axis=1)
target = data['is_ultra']
```

Отделим обучающие данные

In [4]:
```
features_train, features_valid, target_train, target_valid = train_test_split(
    features, target, test_size=0.4, random_state=r_state)
```

Разделим оставшиеся данные на валидационную и тестовую выборки

In [5]:
```
features_valid, features_test, target_valid, target_test = train_test_split(
    features_valid, target_valid, test_size=0.5, random_state=r_state)
```

Проверим правильность разделения выборок

In [6]:
```
features = {'features_train' : features_train, 'features_valid' : features_valid, 'features_test' : features_test}
targets = {'target_train' : target_train, 'target_valid' : target_valid, 'target_test' : target_test}

print('Вспомогательные признаки:')
print()
for i in features:
    sh = features[i].shape
    psh = sh[0]/len(data)
    print('{} - Объектов: {} шт., признаков: {} шт. - {}%'.format(i, sh[0], sh[1], round(psh*100, 2)))

print()

print('Целевые признаки:')
print()
for i in targets:
    sh = targets[i].shape
    psh = sh[0]/len(data)
    print('{} - Объектов: {} шт. - {}%'.format(i, sh[0], round(psh*100, 2)))
```

Вспомогательные признаки:

features_train - Объектов: 1928 шт., признаков: 4 шт. - 59.99%
features_valid - Объектов: 643 шт., признаков: 4 шт. - 20.01%
features_test - Объектов: 643 шт., признаков: 4 шт. - 20.01%

Целевые признаки:

target_train - Объектов: 1928 шт. - 59.99%
target_valid - Объектов: 643 шт. - 20.01%
target_test - Объектов: 643 шт. - 20.01%

Исследуем модели

В нашем исследовании целевой признак - столбец is_ultra (категориальный). Категорий в данном признаке две (тарифы «Смарт» или «Ультра»), соответственно нам предстоит решить задачу бинарной (двоичной) классификации.

Для решения данной задачи используется 3 модели:

- Decision Tree (Дерево решений)
- Random Forest (Случайный лес)
- Logistic Regression (Логистическая регрессия)

Применим все 3 модели, подберем им оптимальные гиперпараметры и найдем лучший результат

Для проверки accuracy на разных моделях создадим функцию:

In [7]:
```
def test_accuracy_score(model):
    predictions_valid = model.predict(features_valid)
    return accuracy_score(target_valid, predictions_valid)
```

Decision Tree (Дерево решений)

In [8]:
```
best_accuracy_DT = 0
best_depth_DT = 0

for depth in range(1, 21):
    model_decision_tree = DecisionTreeClassifier(random_state=r_state, max_depth=depth)
    model_decision_tree.fit(features_train, target_train)
    accuracy = test_accuracy_score(model_decision_tree)
    print('Глубина =', depth, ', accuracy =', accuracy)
    if accuracy > best_accuracy_DT:
        best_accuracy_DT = accuracy
        best_depth_DT = depth

print()
print('Лучший результат:')
print('Глубина =', best_depth_DT, ', accuracy =', best_accuracy_DT)
```

```
Глубина = 1 , accuracy = 0.7542768273716952
Глубина = 2 , accuracy = 0.7822706065318819
Глубина = 3 , accuracy = 0.7853810264385692
Глубина = 4 , accuracy = 0.7791601866251944
Глубина = 5 , accuracy = 0.7791601866251944
Глубина = 6 , accuracy = 0.7838258164852255
Глубина = 7 , accuracy = 0.7822706065318819
Глубина = 8 , accuracy = 0.7791601866251944
Глубина = 9 , accuracy = 0.7822706065318819
Глубина = 10 , accuracy = 0.7744945567651633
Глубина = 11 , accuracy = 0.7620528771384136
Глубина = 12 , accuracy = 0.7620528771384136
Глубина = 13 , accuracy = 0.7558320373250389
Глубина = 14 , accuracy = 0.7589424572317263
Глубина = 15 , accuracy = 0.7465007776049767
Глубина = 16 , accuracy = 0.7340590979782271
Глубина = 17 , accuracy = 0.7356143079315708
Глубина = 18 , accuracy = 0.7309486780715396
Глубина = 19 , accuracy = 0.7278382581648523
Глубина = 20 , accuracy = 0.7216174183514774

Лучший результат:
Глубина = 3 , accuracy = 0.7853810264385692
```

Random Forest (Случайный лес)

In [9]:
```
best_accuracy_RF = 0
best_est_RF = 0
best_depth_RF = 0

for est in range(10, 51, 10):
    for depth in range(1, 21):
        model_random_forest = RandomForestClassifier(random_state=r_state, n_estimators=est, max_depth=depth)
        model_random_forest.fit(features_train, target_train)
```

```
            accuracy = test_accuracy_score(model_random_forest)
            print('Количество деревьев =', est,', глубина =', depth, ', accuracy =', accuracy)
            if accuracy > best_accuracy_RF:
                best_accuracy_RF = accuracy
                best_est_RF = est
                best_depth_RF = depth

    print()
    print('Лучший результат:')
    print('Количество деревьев =', best_est_RF,', глубина =', best_depth_RF, ', accuracy =', best_accuracy_RF)
```

```
Количество деревьев = 10 , глубина = 1 , accuracy = 0.7558320373250389
Количество деревьев = 10 , глубина = 2 , accuracy = 0.7776049766718507
Количество деревьев = 10 , глубина = 3 , accuracy = 0.7853810264385692
Количество деревьев = 10 , глубина = 4 , accuracy = 0.7900466562986003
Количество деревьев = 10 , глубина = 5 , accuracy = 0.7931570762052877
Количество деревьев = 10 , глубина = 6 , accuracy = 0.8009331259720062
Количество деревьев = 10 , глубина = 7 , accuracy = 0.7947122861586314
Количество деревьев = 10 , глубина = 8 , accuracy = 0.7962674961119751
Количество деревьев = 10 , глубина = 9 , accuracy = 0.7853810264385692
Количество деревьев = 10 , глубина = 10 , accuracy = 0.7916018662519441
Количество деревьев = 10 , глубина = 11 , accuracy = 0.7962674961119751
Количество деревьев = 10 , глубина = 12 , accuracy = 0.7884914463452566
Количество деревьев = 10 , глубина = 13 , accuracy = 0.7853810264385692
Количество деревьев = 10 , глубина = 14 , accuracy = 0.7776049766718507
Количество деревьев = 10 , глубина = 15 , accuracy = 0.7776049766718507
Количество деревьев = 10 , глубина = 16 , accuracy = 0.7791601866251944
Количество деревьев = 10 , глубина = 17 , accuracy = 0.7822706065318819
Количество деревьев = 10 , глубина = 18 , accuracy = 0.7776049766718507
Количество деревьев = 10 , глубина = 19 , accuracy = 0.7776049766718507
Количество деревьев = 10 , глубина = 20 , accuracy = 0.7869362363919129
Количество деревьев = 20 , глубина = 1 , accuracy = 0.7667185069984448
Количество деревьев = 20 , глубина = 2 , accuracy = 0.7838258164852255
Количество деревьев = 20 , глубина = 3 , accuracy = 0.7869362363919129
Количество деревьев = 20 , глубина = 4 , accuracy = 0.7884914463452566
Количество деревьев = 20 , глубина = 5 , accuracy = 0.7900466562986003
Количество деревьев = 20 , глубина = 6 , accuracy = 0.7993779160186625
Количество деревьев = 20 , глубина = 7 , accuracy = 0.8009331259720062
Количество деревьев = 20 , глубина = 8 , accuracy = 0.7978227060653188
Количество деревьев = 20 , глубина = 9 , accuracy = 0.7900466562986003
Количество деревьев = 20 , глубина = 10 , accuracy = 0.7916018662519441
Количество деревьев = 20 , глубина = 11 , accuracy = 0.7884914463452566
Количество деревьев = 20 , глубина = 12 , accuracy = 0.8009331259720062
Количество деревьев = 20 , глубина = 13 , accuracy = 0.7931570762052877
Количество деревьев = 20 , глубина = 14 , accuracy = 0.7822706065318819
Количество деревьев = 20 , глубина = 15 , accuracy = 0.7838258164852255
Количество деревьев = 20 , глубина = 16 , accuracy = 0.7900466562986003
Количество деревьев = 20 , глубина = 17 , accuracy = 0.7853810264385692
Количество деревьев = 20 , глубина = 18 , accuracy = 0.7807153965785381
Количество деревьев = 20 , глубина = 19 , accuracy = 0.7838258164852255
Количество деревьев = 20 , глубина = 20 , accuracy = 0.7853810264385692
Количество деревьев = 30 , глубина = 1 , accuracy = 0.7667185069984448
Количество деревьев = 30 , глубина = 2 , accuracy = 0.7838258164852255
Количество деревьев = 30 , глубина = 3 , accuracy = 0.7869362363919129
Количество деревьев = 30 , глубина = 4 , accuracy = 0.7869362363919129
Количество деревьев = 30 , глубина = 5 , accuracy = 0.7931570762052877
Количество деревьев = 30 , глубина = 6 , accuracy = 0.8009331259720062
Количество деревьев = 30 , глубина = 7 , accuracy = 0.80248833592535
Количество деревьев = 30 , глубина = 8 , accuracy = 0.7993779160186625
Количество деревьев = 30 , глубина = 9 , accuracy = 0.7931570762052877
Количество деревьев = 30 , глубина = 10 , accuracy = 0.7947122861586314
Количество деревьев = 30 , глубина = 11 , accuracy = 0.7853810264385692
Количество деревьев = 30 , глубина = 12 , accuracy = 0.7947122861586314
Количество деревьев = 30 , глубина = 13 , accuracy = 0.7869362363919129
Количество деревьев = 30 , глубина = 14 , accuracy = 0.7822706065318819
Количество деревьев = 30 , глубина = 15 , accuracy = 0.7853810264385692
Количество деревьев = 30 , глубина = 16 , accuracy = 0.7931570762052877
Количество деревьев = 30 , глубина = 17 , accuracy = 0.7853810264385692
Количество деревьев = 30 , глубина = 18 , accuracy = 0.7776049766718507
Количество деревьев = 30 , глубина = 19 , accuracy = 0.7838258164852255
Количество деревьев = 30 , глубина = 20 , accuracy = 0.7838258164852255
Количество деревьев = 40 , глубина = 1 , accuracy = 0.776049766718507
Количество деревьев = 40 , глубина = 2 , accuracy = 0.7853810264385692
Количество деревьев = 40 , глубина = 3 , accuracy = 0.7869362363919129
Количество деревьев = 40 , глубина = 4 , accuracy = 0.7900466562986003
Количество деревьев = 40 , глубина = 5 , accuracy = 0.7947122861586314
Количество деревьев = 40 , глубина = 6 , accuracy = 0.80248833592535
Количество деревьев = 40 , глубина = 7 , accuracy = 0.80248833592535
Количество деревьев = 40 , глубина = 8 , accuracy = 0.8087091757387247
Количество деревьев = 40 , глубина = 9 , accuracy = 0.7947122861586314
Количество деревьев = 40 , глубина = 10 , accuracy = 0.7962674961119751
Количество деревьев = 40 , глубина = 11 , accuracy = 0.7853810264385692
Количество деревьев = 40 , глубина = 12 , accuracy = 0.7931570762052877
```

```
Количество деревьев = 40 , глубина = 13 , accuracy = 0.7838258164852255
Количество деревьев = 40 , глубина = 14 , accuracy = 0.7853810264385692
Количество деревьев = 40 , глубина = 15 , accuracy = 0.7838258164852255
Количество деревьев = 40 , глубина = 16 , accuracy = 0.7931570762052877
Количество деревьев = 40 , глубина = 17 , accuracy = 0.7916018662519441
Количество деревьев = 40 , глубина = 18 , accuracy = 0.7776049766718507
Количество деревьев = 40 , глубина = 19 , accuracy = 0.7838258164852255
Количество деревьев = 40 , глубина = 20 , accuracy = 0.7838258164852255
Количество деревьев = 50 , глубина = 1 , accuracy = 0.7589424572317263
Количество деревьев = 50 , глубина = 2 , accuracy = 0.7838258164852255
Количество деревьев = 50 , глубина = 3 , accuracy = 0.7869362363919129
Количество деревьев = 50 , глубина = 4 , accuracy = 0.7869362363919129
Количество деревьев = 50 , глубина = 5 , accuracy = 0.7931570762052877
Количество деревьев = 50 , глубина = 6 , accuracy = 0.7993779160186625
Количество деревьев = 50 , глубина = 7 , accuracy = 0.80248833592535
Количество деревьев = 50 , глубина = 8 , accuracy = 0.807153965785381
Количество деревьев = 50 , глубина = 9 , accuracy = 0.7978227060653188
Количество деревьев = 50 , глубина = 10 , accuracy = 0.7931570762052877
Количество деревьев = 50 , глубина = 11 , accuracy = 0.7900466562986003
Количество деревьев = 50 , глубина = 12 , accuracy = 0.7962674961119751
Количество деревьев = 50 , глубина = 13 , accuracy = 0.7884914463452566
Количество деревьев = 50 , глубина = 14 , accuracy = 0.7916018662519441
Количество деревьев = 50 , глубина = 15 , accuracy = 0.7838258164852255
Количество деревьев = 50 , глубина = 16 , accuracy = 0.7916018662519441
Количество деревьев = 50 , глубина = 17 , accuracy = 0.7947122861586314
Количество деревьев = 50 , глубина = 18 , accuracy = 0.7869362363919129
Количество деревьев = 50 , глубина = 19 , accuracy = 0.7900466562986003
Количество деревьев = 50 , глубина = 20 , accuracy = 0.7853810264385692

Лучший результат:
Количество деревьев = 40 , глубина = 8 , accuracy = 0.8087091757387247
```

Logistic Regression (Логистическая регрессия)

In [10]:
```
best_accuracy_LR = 0
best_solver_LR = ''
solvers = ['lbfgs', 'liblinear', 'newton-cg', 'sag', 'saga']

for solver in solvers:
    model_logistic_regression = LogisticRegression(solver=solver, random_state=r_state)
    model_logistic_regression.fit(features_train, target_train)
    accuracy = test_accuracy_score(model_logistic_regression)
    print('Алгоритм =', solver, ', accuracy =', accuracy)
    if accuracy > best_accuracy_LR:
        best_accuracy_LR = accuracy
        best_solver_LR = solver

print()
print('Лучший результат:')
print('Алгоритм =', best_solver_LR, ', accuracy =', best_accuracy_LR)
```

```
Алгоритм = lbfgs , accuracy = 0.7107309486780715
Алгоритм = liblinear , accuracy = 0.7542768273716952
c:\ProgramData\Anaconda3\lib\site-packages\scipy\optimize\_linesearch.py:456: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
c:\ProgramData\Anaconda3\lib\site-packages\scipy\optimize\_linesearch.py:305: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
c:\ProgramData\Anaconda3\lib\site-packages\scipy\optimize\_linesearch.py:456: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
c:\ProgramData\Anaconda3\lib\site-packages\scipy\optimize\_linesearch.py:305: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
c:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_sag.py:352: ConvergenceWarning: The max_iter was reached which means the coef_ di
d not converge
  warnings.warn(
Алгоритм = newton-cg , accuracy = 0.7558320373250389
Алгоритм = sag , accuracy = 0.7060653188180405
Алгоритм = saga , accuracy = 0.7060653188180405

Лучший результат:
Алгоритм = newton-cg , accuracy = 0.7558320373250389
c:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_sag.py:352: ConvergenceWarning: The max_iter was reached which means the coef_ di
d not converge
  warnings.warn(
```

Определим лучшую модель

In [11]:
```
max_accuracy = best_accuracy_DT
if max_accuracy < best_accuracy_RF:
    max_accuracy = best_accuracy_RF
if max_accuracy < best_accuracy_LR:
    max_accuracy = best_accuracy_LR

if max_accuracy == best_accuracy_DT:
```

```
    print('Лучшая модель: Decision Tree (Дерево решений), с гиперпараметром max_depth =', best_depth_DT)
if max_accuracy == best_accuracy_RF:
    print('Лучшая модель: Random Forest (Случайный лес), с гиперпараметрами n_estimators =', best_est_RF,
      'max_depth =', best_depth_RF)
if max_accuracy == best_accuracy_LR:
    print('Лучшая модель: Logistic Regression (Логистическая регрессия), с гиперпараметром solver =', best_solver_LR)
print('accuracy =', max_accuracy)
```

Лучшая модель: Random Forest (Случайный лес), с гиперпараметрами n_estimators = 40 max_depth = 8
accuracy = 0.8087091757387247

Проверим модель на тестовой выборке

Объединим обучающую и валидационную выборки для обучения итоговой модели на большем количестве данных

In [12]:
```
features = pd.concat([features_train, features_valid])
target = pd.concat([target_train, target_valid])
```

Обучим итоговую модель и проверим результат на тестовой выборке

In [13]:
```
model = RandomForestClassifier(random_state=r_state, n_estimators=best_est_RF, max_depth=best_depth_RF)
model.fit(features, target)
predictions_test = model.predict(features_test)
model_accuracy = accuracy_score(target_test, predictions_test)
print('accuracy =', model_accuracy)
```

accuracy = 0.7993779160186625
Результат accuracy на тестовой выборке (0.7993) не превышаает и близок к результату на валидационной выборке (0.8087), значит мы избежали проблемы переобучения. Так же результат больше 0.75 что означает отсутвие проблемы недообучения

Проверим модели на адекватность

Так как перед нами стоит задача классификации, то для проверки на адекватность нашей итоговой модели необходимо сравнить ее с моделью, которая все время предсказывает один класс. Для этого создадим и обучим сравнительную модель с помощью DummyClassifier с гиперпараметром strategy='most_frequent' (всегда возвращает наиболее часто встречающийся класс)

In [14]:
```
model_dummy = DummyClassifier(random_state=r_state, strategy='most_frequent')
model_dummy.fit(features, target)
predictions_test_dummy = model_dummy.predict(features_test)
model_dummy_accuracy = accuracy_score(target_test, predictions_test_dummy)
print('accuracy =', model_dummy_accuracy)
```

accuracy = 0.6842923794712286
Сравним результаты и оценим адекватность нашей итоговой модели

In [15]:
```
if model_accuracy > model_dummy_accuracy:
    print('Итоговая модель является адекватной')
else:
    print('Итоговая модель является неадекватной')
```

Итоговая модель является адекватной

Общий вывод

Проведено исследование для поиска модели с максимально большим значением *accuracy* (не меньше 0.75)

Данные о поведении клиентов были взяты из файла users_behavior.csv

Исследование проходило в пять этапов:

- Обзор и изучение данных
  - Предобработка данных выполнена ранее
  - Данные состоят из 3214 объектов
  - Данные имеют 5 признаков (1 целевой и 4 вспомогательных)
- Разделение данных на выборки
  - Определили целевой признак (is_ultra)
  - Разделили данные на 3 выборки:
    - Обучающую (60%)
    - Валидационную (20%)
    - Тестовую (20%)
- Исследование различных моделей и подбор лучших гиперпараметров
  - Decision Tree (Дерево решений) - Лучший результат: Глубина = 3 , accuracy = 0.7853810264385692
  - Random Forest (Случайный лес) - Лучший результат: Количество деревьев = 40 , глубина = 8 , accuracy = 0.8087091757387247
  - Logistic Regression (Логистическая регрессия) - Лучший результат: Алгоритм = newton-cg , accuracy = 0.7558320373250389
- Проверка итоговой модели на тестовой выборке
  - Объединили обучающую и валидационную выборки для обучения итоговой модели на большем количестве данных
  - Результат accuracy на тестовой выборке (0.7993)
  - Результат на тестовой выборке не превышаает и близок к результату на валидационной выборке (0.8087), значит мы избежали проблемы переобучения
  - Результат на тестовой выборке больше 0.75 что означает отсутвие проблемы недообучения
- Проверка модели на адекватность
  - Сравнили итоговую модель с моделью, которая все время предсказывает один класс
  - Обучили сравнительную модель с помощью DummyClassifier с гиперпараметром strategy='most_frequent' (всегда возвращает наиболее часто встречающийся класс)
  - Результат accuracy сравнительной модели = 0.6842923794712286
  - Результат accuracy нашей итоговой модели значительно больше результата сравнительной модели
  - Итоговая модель является адекватной