

Привет, меня зовут Александр Куимов. Я буду ревьюером твоего проекта. Ты можешь обращаться ко мне на "ты" 😊 Надеюсь, тебя также не смутит, если я буду обращаться к тебе на "ты", но если это неудобно, обязательно скажи об этом!

Пожалуйста, не удаляй мои комментарии, они будут особенно полезны для нашей работы в случае повторной проверки проекта.

Ты также можешь реагировать на мои комментарии своими по шаблону, показанному чуть ниже. Это нужно, чтобы не создавалась путаница 😊

Ты можешь найти мои комментарии, обозначенные **зеленым**, **желтым** и **красным** цветами, например:

Комментарий ревьюера

Все отлично! 🙌: В случае, если решение на отдельном шаге является полностью правильным.

Комментарий ревьюера

Некоторые замечания и рекомендации 💡: В случае, когда решение на отдельном шаге станет еще лучше, если внести небольшие коррективы.

Комментарий ревьюера

На доработку 😞: В случае, когда решение на отдельном шаге требует существенной переработки и внесения правок. Напоминаю, что проект не может быть принят с первого раза, если ревью содержит комментарии, рекомендующие доработать шаги.

Комментарий студента:

👤: В такой цветовой ячейке я прошу тебя оставлять свои комментарии. Если исправляешь проект на второй итерации и выше, не забывай пожалуйста указывать номер итерации, например, "Комментарий студента v.2".

Увидев у тебя неточность, в первый раз я лишь укажу на ее наличие и дам тебе возможность самому найти и исправить ее. На реальной работе твой руководитель будет поступать также, и я пытаюсь подготовить тебя именно к работе датасаентистом. Но если ты пока не справишься с такой задачей - при следующей проверке я дам более точную подсказку! 🤖

Рекомендация тарифов

✓ Описание проекта

Цели проекта

Многие клиенты оператора мобильной связи пользуются архивными тарифами. Есть данные о поведении клиентов, которые уже перешли на новые тарифы. Необходимо построить систему, способную проанализировать поведение клиентов и исходя из их поведения предложить пользователям новый тариф: «Смарт» или «Ультра». Нужно построить модель для задачи классификации, которая выберет подходящий тариф.

Задач проекта

- Постройте модель с максимально большим значением ассурасу.
- Довести долю правильных ответов по крайней мере до 0.75.
- Проверьте ассурасу на тестовой выборке.

Заказчик проекта Оператор мобильной связи «Мегалайн»

Входные данные:

Каждый объект в наборе данных — это информация о поведении одного пользователя за месяц. Известно:

- calls — количество звонков,
- minutes — суммарная длительность звонков в минутах,
- messages — количество sms-сообщений,
- mb_used — израсходованный интернет-трафик в Мб,
- is_ultra — каким тарифом пользовался в течение месяца («Ультра» — 1, «Смарт» — 0).

План исследования:

- 1) Ознакомиться с данными (подготовка данных не понадобится - она уже проведена)
- 2) Разбить данные на выборки:
 - обучающую
 - валидационную
 - тестовую
- 3) Исследовать модели с различными гиперпараметрами, расчет ассигару для каждой модели:
 - решающее дерево с глубиной ветвления дерева
 - случайный лес: количество деревьев и глубина ветвления дерева
 - логистическая регрессия
- 4) Проверить модели на тестовой выборке
- 5) Проверить модель на адекватность
- 6) Общие выводы

Комментарий ревьюера

Все отлично!👍:

Вижу твоё добавленное описание проекта. Молодец! Это поможет тебе расставлять акценты в выводах


Откройте и изучите файл

```
#подключаем библиотеки
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```


```
df = pd.read_csv('/datasets/users_behavior.csv')
```

```
df.head()
```



	calls	minutes	messages	mb_used	is_ultra
0	40.0	311.90	83.0	19915.42	0
1	85.0	516.75	56.0	22696.96	0
2	77.0	467.66	86.0	21060.45	0
3	106.0	745.53	81.0	8437.39	1
4	66.0	418.74	1.0	14502.75	0

```
df.info()
```



```
<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
```

Выводы: пропусков в данных нет, данные готовы для анализа и построения модели классификации и рекомендации нового тарифа. Объем данных 3214 строк

Комментарий ревьюера

Все отлично! 🍌: С данными познакомились, проведен первичный аналитический осмотр!) Полученная информация поможет нам в дальнейшем с предобработкой данных.

Рекомендую также тебе посмотреть метод `sns.pairplot` с параметром `hue='is_ultra'`, в который мы передаем целевой признак.

✓ Разбейте данные на выборки

Для создания модели, оценки метрики accuracy, а так же проверки работы модели необходимы три выборки данных. У нас есть одна. Ее и разобьем на три в соотношении 3:1:1:

- обучающая
- валидационная
- тестовая

В задачах моделирования удобно целевой признак хранить отдельно от остальных признаков. Разделим данные и посмотрим на объемы полученных выборок

```
features = df.drop('is_ultra', axis=1) # отделяем признаки
target = df['is_ultra'] # отделяем целевой признак

# выделяем валидационную выборку features_valid и target_valid
features_tr, features_valid, target_tr, target_valid = train_test_split(
    features, target, test_size=0.20, random_state=12345)
# выделяем тестовую выборку features_test и target_test и оставшиеся данные - данные для обучения features_train target_train
features_train, features_test, target_train, target_test = train_test_split(
    features_tr, target_tr, test_size=0.25, random_state=12345)

print(f'Размер {len(features_train)/len(df):.2} обучающей выборки: ')
print(features_train.shape)
print(target_train.shape)
print(f'Размер {len(features_valid)/len(df):.2} валидационной выборки:')
print(features_valid.shape)
print(target_valid.shape)
print(f'Размер {len(features_test)/len(df):.2} тестовой выборки:')
print(features_test.shape)
print(target_test.shape)
```

```
→ Размер 0.6 обучающей выборки:
(1928, 4)
(1928,)
Размер 0.2 валидационной выборки:
(643, 4)
(643,)
Размер 0.2 тестовой выборки:
(643, 4)
(643,)
```

Выводы и результаты: целевой признак отделен от признаков. Данные разбиты на обучающую, валидационную и тестовую выборки. Объемы выборок 1928:643:643 строк

Комментарий ревьюера

Все отлично! 🍌:

С разбиением все в порядке, а размеры полученных наборов напечатаны)

✓ Исследуйте модели

✓ Модель решающее дерево

Обучим модель решающее дерево. Обучим модели с глубиной ветвления дерева от 1 до 10 и посмотрим метрику accuracy для каждой модели.

Создадим структуру данных, в которую будем записывать результаты для всех моделей в проекте: `df_models_results`

```
#модель решающее дерево
#гиперпараметр глубина дерева
model_dtrees = DecisionTreeClassifier(random_state=12345)
model_dtrees.fit(features_train, target_train)
predictions_valid = model_dtrees.predict(features_valid)
```

```

best_deth_dtree = 0
best_result_dtree = 0
best_model_dtree = None
df_models_results = pd.DataFrame()

# цикл для max_depth от 1 до 10 >
for i in range(1,11):
    model_dtree = DecisionTreeClassifier(max_depth=i, random_state=12345)
    # обучение модели
    model_dtree.fit(features_train, target_train)
    # предсказания модели на валидационной выборке
    predictions_valid=model_dtree.predict(features_valid)
    # расчет доли верных предсказаний к размеру выборки
    result = accuracy_score(target_valid, predictions_valid)
    print(f'max_depth = {i} Доля верных предсказаний:{ result:.3}')
    if (best_result_dtree < result):
        best_result_dtree = result
        best_deth_dtree = i
        best_model_dtree = model_dtree

print()
print(f'Глубина дерева для наилучшей модели max_depth = {best_deth_dtree} Доля верных предсказаний:{ best_result_dtree:.3}')

# запишем результаты по первой модели
if (len(df_models_results.index)<1):
    df_models_results = df_models_results.append(
        {
            'model_name': "решающее дерево",
            'best_depth': best_deth_dtree,
            'best_est': None,
            'best_result': round(best_result_dtree,3)}, ignore_index=True)
df_models_results

```

```

→ max_depth = 1 Доля верных предсказаний:0.748
max_depth = 2 Доля верных предсказаний:0.784
max_depth = 3 Доля верных предсказаний:0.787
max_depth = 4 Доля верных предсказаний:0.787
max_depth = 5 Доля верных предсказаний:0.788
max_depth = 6 Доля верных предсказаний:0.779
max_depth = 7 Доля верных предсказаний:0.788
max_depth = 8 Доля верных предсказаний:0.781
max_depth = 9 Доля верных предсказаний:0.778
max_depth = 10 Доля верных предсказаний:0.771

```

Глубина дерева для наилучшей модели max_depth = 5 Доля верных предсказаний:0.788

	best_depth	best_est	best_result	model_name
0	5.0	None	0.788	решающее дерево

Вывод: максимальная доля верных предсказаний 0,788 у модели с глубиной ветвления дерева 5.

Комментарий ревьюера

Все отлично! 🍌: Метрика выбрана верно!

✓ Случайный лес

Обучим модель случайный лес. Исследуем влияние двух гиперпараметров на предсказания модели: глубина ветвления дерева и количество деревьев.

```
#модель случайный лес
best_model_forest = None
best_result_forest = 0
best_depth_forest=0
best_est_forest=0

for est in range(10, 51, 10):
    for depth in range(1, 11):
        model_forest = RandomForestClassifier(random_state=12345, n_estimators=est, max_depth=depth) # обучите модель с заданным количеством деревьев
        model_forest.fit(features_train,target_train) # обучите модель на тренировочной выборке
        result = model_forest.score(features_valid, target_valid) # посчитайте качество модели на валидационной выборке
        if result > best_result_forest:
            best_model_forest = model_forest# сохраните наилучшую модель
            best_result_forest = result# сохраните наилучшее значение метрики accuracy на валидационных данных
            best_est_forest = est
            best_depth_forest = depth
#выведем только 5 результатов, перебираемых в двух циклах
print(f"Доля верных предсказаний модели на валидационной выборке:{ result:.3} Количество деревьев: {est} Максимальная глубина: {depth}")

print()
print(f"Доля верных предсказаний у наилучшей модели на валидационной выборке: {best_result_forest:.3} Количество деревьев: {best_est_forest:.3} Максимальная глубина: {best_depth_forest:.3}")

# запишем результаты по второй модели
if (len(df_models_results.index)<2):
    df_models_results = df_models_results.append({'model_name': "случайный лес",
                                                    'best_depth': best_depth_forest,
                                                    'best_est': best_est_forest,
                                                    'best_result': round(best_result_forest,3)}, ignore_index=True)
```

df_models_results

Доля верных предсказаний модели на валидационной выборке:0.788 Количество деревьев: 1
 Доля верных предсказаний модели на валидационной выборке:0.788 Количество деревьев: 2
 Доля верных предсказаний модели на валидационной выборке:0.799 Количество деревьев: 3
 Доля верных предсказаний модели на валидационной выборке:0.801 Количество деревьев: 4
 Доля верных предсказаний модели на валидационной выборке:0.799 Количество деревьев: 5

Доля верных предсказаний у наилучшей модели на валидационной выборке: 0.801 Количество деревьев: 20 Максимальная глубина: 6

	best_depth	best_est	best_result	model_name
0	5.0	None	0.788	решающее дерево
1	6.0	20	0.801	случайный лес

Выводы:

- максимальная доля верных предсказаний у модели случайный лес 0,801 с гиперпараметрами:
 - глубина ветвления деревьев - 6,
 - количество деревьев 20
- доля верных предсказаний выше, чем у модели решающее дерево
- время, чтобы найти наилучшую модель среди моделей случайного леса несоизмеримо больше, чем для решающего дерева

Комментарий ревьюера

Все отлично! 🍌: Хорошо, что есть промежуточный вывод)

✓ Логистическая регрессия

#Логистическая регрессия

```
model_logreg = LogisticRegression(random_state=12345, solver='lbfgs', max_iter=1000)
model_logreg.fit(features_train, target_train)
predictions_valid=model_logreg.predict(features_valid)
best_result_logreg = model_logreg.score(features_valid, target_valid)
print(f'Доля верных предсказаний:{ best_result_logreg:.3}')

# запишем результаты по третьей модели
if (len(df_models_results.index)<3):
    df_models_results = df_models_results.append({'model_name': "логистическая регрессия",
                                                    'best_depth': None,
                                                    'best_est': None,
                                                    'best_result': round(best_result_logreg,3)}, ignore_index=True)

df_models_results
```

Доля верных предсказаний: 0.759

	best_depth	best_est	best_result	model_name
0	5.0	None	0.788	решающее дерево
1	6.0	20	0.801	случайный лес
2	None	None	0.759	логистическая регрессия

Вывод: модель логистической регрессии работает очень быстро, но ее доля верных предсказаний на валидационной выборке в данном исследовании оказалась наименьшей, но все же выше 0,75 = 0,759

Комментарий ревьюера

Все отлично! 👍:

Модели исследованы корректно:

- исследовано более 5 значений гиперпараметра
- модели обучены на обучающем наборе
- получена оценка качества на валидационном наборе
- перебор гиперпараметров осуществляется в цикле
- есть выводы

✓ Проверьте модель на тестовой выборке

```
#решающее дерево
predictions_test = best_model_dtrees.predict(features_test)
result_dtrees_test = accuracy_score(target_test, predictions_test)

print('Модель решающее дерево')
print(f'Доля верных предсказаний на валидационной выборке:{ best_result_dtrees:.3}')
```

Модель решающее дерево
Доля верных предсказаний на валидационной выборке: 0.788
Доля верных предсказаний на тестовой выборке: 0.759

Доля верных ответов на тестовой выборке ниже, чем на валидационной 75,9% < 78,8%. Но все же больше 75%.

```
#случайный лес
result_forest_test = best_model_forest.score(features_test, target_test)
print('Модель случайный лес')
print(f'Доля верных предсказаний на валидационной выборке:{ best_result_forest:.3}')
```

Модель случайный лес
Доля верных предсказаний на валидационной выборке: 0.801
Доля верных предсказаний на тестовой выборке: 0.785

Вывод: доля верных предсказаний на тестовой выборке для модели случайный лес так же ниже, чем на валидационной выборке 78,5% < 80,1%, но выше чем для наилучшей модели решающего дерева 78,5% > 75,9%

Комментарий ревьюера

Некоторые замечания и рекомендации 💡:

На тестовой выборке получено хорошее качество (она у нас играет роль отложенной, holdout), но знай, что тестирование положено проводить для одной лучшей модели. На предыдущем шаге мы должны были настроить модели и выбрать одну наилучшую, опираясь на метрики данные заказчиком (это может быть качество, время обучения, скорость предсказания и т.д.). Тестированием мы моделируем работу модели на новых незнакомых ей данных, которые ни разу не использовались ни при тренировке, ни при валидации, и проверяем, не словили ли мы переобучение. А эти данные могут быть смещенными, с выбросами и т.д. То есть по таким данным некорректно заново переопределять модель-победитель. Советую тебе статью, рассматривающую разные способы валидации моделей машинного обучения: <https://towardsdatascience.com/validating-your-machine-learning-model-25b4c8643fb7> (нужен VPN для открытия)

Метрики качества на тестовой выборке для всех моделей можно вычислить только с целью исследования их смещения относительно аналогичных метрик на валидационной выборке. Но это не означает, что модель-победитель должна выбираться исходя из сравнения метрик, полученных на тестовой выборке.

```
#логистическая регрессия
result_logreg_test = model_logreg.score(features_test, target_test)
print('Модель логистическая регрессия')
print(f'Доля верных предсказаний на валидационной выборке:{ best_result_logreg:.3}')
print(f'Доля верных предсказаний на тестовой выборке:{ result_logreg_test:.3}')
```

➡ Модель логистическая регрессия
Доля верных предсказаний на валидационной выборке:0.759
Доля верных предсказаний на тестовой выборке:0.726

Вывод: доля верных предсказаний на тестовой выборке меньше и ниже 75%, но модель работает быстрее всех остальных. На тестовой 72,6%, на валидационной 75,9% доля верных предсказаний

Комментарий ревьюера

Все отлично! 🍌: Хорошо, что есть промежуточный вывод)

Выведем все результаты вместе

```
#вставить столбец с тестовыми результатами
df_models_results.insert(loc= len(df_models_results.columns) , column='test_result',
                          value=[ round(result_dtree_test,3), round(result_forest_test,3), round(result_logreg_test,3)])
df_models_results
```

➡

	best_depth	best_est	best_result	model_name	test_result
0	5.0	None	0.788	решающее дерево	0.759
1	6.0	20	0.801	случайный лес	0.785
2	None	None	0.759	логистическая регрессия	0.726

Вывод: для всех моделей доля верно предсказанных ответов на тестовой выборке меньше, чем на валидационной. Наблюдается переобучение моделей.

Комментарий ревьюера

Все отлично! 🍌: Хорошо, что есть промежуточный вывод)

✓ (бонус) Проверьте модели на адекватность

Допустим у нас случайная модель. Она предсказывает тарифы случайно - случайным образом. Тогда ассигасу такой модели считаются следующим образом: для тарифа 0 вероятность правильных ответов 50%, для тарифа 1 вероятность 50% правильно попасть. Тогда для случайной модели ассигасу = $0,25 + 0,25 = 0,5$

Тогда, все модели рассмотренные выше имеют ассигасу существенно выше, чем для случайной модели и они адекватны.

Комментарий ревьюера

Все отлично! 🍌:

Этап тестирования рекомендуется дополнять введением в работу baseline модели. Сравнивая результаты более сложных моделей с самой наивной, мы можем делать выводы о том, насколько далеко они вообще продвинулись в качестве предсказаний. Иногда может случиться такое, что константная модель оказывается не сильно хуже по качеству. Это может говорить либо о том, что еще есть смысл продолжить поиски "той самой" модели, либо структура данных такова, что ничего лучше константоной в принципе не найти, тогда в использовании более сложных ресурсозатратных моделей нет особого смысла. Создавать константные модели можно либо вручную, либо воспользоваться готовым алгоритмом [DummyClassifier](#) из пакета `sklearn`. В нашем случае подойдет модель `DummyClassifier` со стратегией `most_frequent`. Рекомендую также прочитать [статью](#).

✓ Общий вывод

В рамках исследования были проведены следующие шаги:

1. Загружены данные
2. Данные разделены на выборки:
 - обучающую
 - валидационную
 - тестовую
3. Обучены три модели:
 - решающее дерево
 - случайный лес
 - логистическая регрессия
4. Исследованы модели с разными гиперпараметрами
5. Проверена работа моделей на тестовых выборках
6. Проверена адекватность моделей

В ходе проекта установлены следующие факты:

Пропусков в данных нет, данные готовы для анализа и построения модели классификации и рекомендации нового тарифа. Объем данных 3214 строк

Целевой признак отделен от остальных признаков. Данные разбиты на обучающую, валидационную и тестовую выборки. Объемы выборок 1928:643:643 строк.

Обученные модели решающее дерево с глубиной ветвления дерева от 1 до 10 показали значения метрики accuracy, от 0,748 – для $\text{max_depth}=1$, до 0,78 $\text{max_depth}=5$ и снова падает до 0,771 $\text{max_depth}=10$. Максимальная доля верных предсказаний 0,788 у модели с глубиной ветвления дерева 5.

Для модели случайный лес доля верных предсказаний у наилучшей модели на валидационной выборке: 0.801

- Количество деревьев: 20
- Максимальная глубина: 6

Модель логистической регрессии работает очень быстро, но ее доля верных предсказаний на валидационной выборке в данном исследовании оказалась наименьшей, но все же выше 0,75: 0,759

Проверка на тестовой выборке показала следующие результаты:

Модель решающее дерево $\text{max_depth}=5$:

- дала долю верных ответов на тестовой выборке ниже, чем на валидационной 75,9% < 78.8%. Но все же больше 75%.

Модель случайный лес с $\text{max_depth}=6$, $\text{n_estimators}=20$ деревьев:

- Доля верных предсказаний на валидационной выборке: 0.801
- Доля верных предсказаний на тестовой выборке: 0.785
- Доля верных предсказаний на тестовой выборке для модели случайный лес так же ниже, чем на валидационной выборке 78,5% < 80,1%, но выше чем для наилучшей модели решающего дерева 78,5% > 75,9%

Модель логистическая регрессия:

- Доля верных предсказаний на валидационной выборке: 0.759
- Доля верных предсказаний на тестовой выборке: 0.726
- Доля верных предсказаний на тестовой выборке меньше и к тому же ниже 75%. На тестовой 72,6%, на валидационной 75,9% доля верных предсказаний

Если сравнивать модели со случайным предсказанием тарифов, то все модели дают адекватные результаты.

Вывод: самые лучшие результаты по предсказанию тарифов показала модель случайный лес с гиперпараметрами: Количество деревьев: 20 Максимальная глубина: 6.

Итоговый комментарий ревьюера

У меня сложилось хорошее впечатление о проекте. Молодец! Изучены все параметры, построено несколько моделей классификации и оценено их качество. Осмысленная аналитика и дельная модельная работа - многое удалось как надо)

Отмечу отдельные положительные моменты проекта 😊:

- тебе удалось добиться очень хорошего качества, поздравляю!

- при обучении моделей использована валидационная выборка и подбор гиперпараметров.

Проект может быть зачтен, но я его отправлю назад, чтобы у тебя была возможность задать вопросы и внести правки, при желании. Однако, ты можешь просто вернуть проект в таком же виде и я его зачту.

✓ Чек-лист готовности проекта

Поставьте 'x' в выполненных пунктах. Далее нажмите Shift+Enter.

- ☒ Jupyter Notebook открыт
- ☒ Весь код выполняется без ошибок
- ☒ Ячейки с кодом расположены в порядке исполнения
- ☒ Выполнено задание 1: данные загружены и изучены
- ☒ Выполнено задание 2: данные разбиты на три выборки
- ☒ Выполнено задание 3: проведено исследование моделей
 - ☒ Рассмотрено больше одной модели
 - ☒ Рассмотрено хотя бы 3 значения гиперпараметров для какой-нибудь модели
 - ☒ Написаны выводы по результатам исследования
- ☒ Выполнено задание 3: Проведено тестирование
- ☒ Удалось достичь ассигасы не меньше 0.75