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.

Откройте и изучите файл

In [31]:

```
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
import seaborn as sns
from sklearn.model_selection import GridSearchCV
```

In [32]:

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

In [33]:

```
df.head()
```

Out[33]:

	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

In [34]:

```
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
```

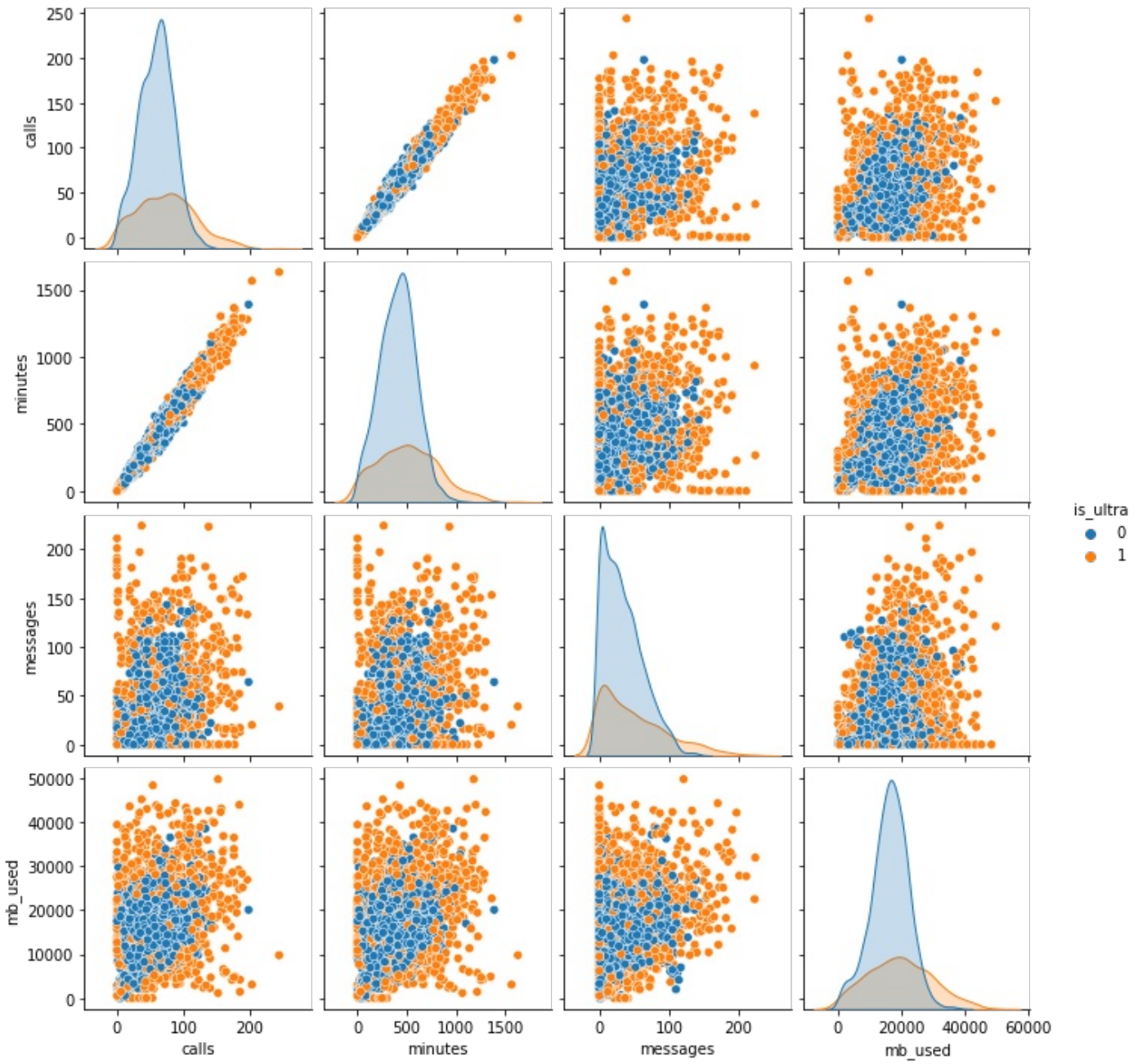
Отлично, все значения являются числами, а признак `is_ultra` является целочисленным.

In [35]:

```
sns.pairplot(df, hue='is_ultra')
```

Out[35]:

```
<seaborn.axisgrid.PairGrid at 0x7efd53bf8370>
```



Сразу бросается в глаза сильная корреляция между звонками и минутами. Если мы удалим один из признаков, это может положительно сказаться на эффективности моделей

Разбейте данные на выборки

In [36]:

```
# Выделяю признаки и целевой признак
```

```
features = df.drop(['is_ultra'], axis=1)
target = df['is_ultra']
```

In [37]:

```
# Делю признаки и целевой признак на тренировочную и тестовую (где тестовая 20% от данных
)
features_train, features_test, target_train, target_test = train_test_split(features, tar
get,
                                                                            test_size=.
4)
# Делю уже тренировочную
features_test, features_valid, target_test, target_valid = train_test_split(features_test
,
                                                                            target_te
st,
                                                                            test_size
=.5)
```

In [38]:

```
# Проверим, правильно ли все поделили
print(len(features))
print(len(features_train))
print(len(features_test))
print(len(features_valid))
```

```
3214

1928

643

643
```

Данные делили отношением **3:1:1,** по выведенным значениям все верно.

Исследуйте модели

Поочередно исследуем наилучшие модели, меняя гиперпараметры.

In [39]:

```
# Исследование модели по глубине делева

best_model = None
best_result = 0
for depth in range(1, 6):
    model = DecisionTreeClassifier(random_state=12345, max_depth=depth) # обучение модел
ь с заданной глубиной дерева
    model.fit(features_train, target_train) # обучаем модель
    predictions = model.predict(features_valid) # получаем предсказания модели
    result = accuracy_score(target_valid, predictions) # считаем качество модели
    if result > best_result:
        best_model = model
        best_result = result
        depth = depth

print("Accuracy лучшей модели:", best_result)
print('Глубина дерева:', depth)
```

```
Accuracy лучшей модели: 0.8118195956454122

Глубина дерева: 5
```

In [40]:

```
# Исследование по кол-ву деревьев
```

```
best_model = None
best_result = 0
for est in range(1, 11):
    model = RandomForestClassifier(random_state=12345, n_estimators=est)
    model.fit(features_train, target_train)
    result = model.score(features_valid, target_valid)
    if result > best_result:
        best_model = model
        best_result = result
        est = est

print("Accuracy лучшей модели:", best_result)
print("Количество деревьев:", est)
```

```
Accuracy лучшей модели: 0.8180404354587869

Количество деревьев: 10
```

In [41]:

```
# Расчет логистической регрессии

best_model = None
best_result = 0
for iter in range(100, 1001, 100):
    model = LogisticRegression(random_state=12345, solver='lbfgs', max_iter=1000)
    model.fit(features_train, target_train)
    result = model.score(features_valid, target_valid)
    if result > best_result:
        best_model = model
        best_result = result
        iter = iter

print('Accuracy лучшей модели:', best_result)
print('Количество итераций обучения:', iter)
```

```
Accuracy лучшей модели: 0.7013996889580093

Количество итераций обучения: 1000
```

In [45]:

```
# Эксперимент с GridSearchCV

clf = RandomForestClassifier()

parameters = {'n_estimators': range(1, 11, 1),
              'max_depth': range(1, 6, 1)}

grid = GridSearchCV(clf, parameters, cv=5)
grid.fit(features_train, target_train)

grid.best_score_
```

Out[45]:

```
0.8018531727340017
```

Проверьте модель на тестовой выборке

Мы узнали, что наилучшая модель считается по случайному лесу, где оптимальное кол-во деревьев равно **10.** Будем тестировать эту модель.

In [46]:

```
model = RandomForestClassifier(random_state=12345, n_estimators=10)
model.fit(features_train, target_train)
print('Accuracy случайного леса на тестовой выборке:', model.score(features_test, target_
```

```
test))
```

```
Accuracy случайного леса на тестовой выборке: 0.7698289269051322
```

Ура**,** значение `accuracy` больше **0,75.**

Вывод: модель случайного леса показала себя лучше остальных на валидационной выборке и после достигла неплохого результата на тестовой**.** Это говорит о том**,** что модель не переобучена и недообучена**.**

Чек-лист готовности проекта

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

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

In [ ]:

```

```