Рубежный контроль №2

Вариант №10

Выполнил: Ли М.В.

Группа: ИУ5-64Б

Задание:

Линейная/логистическая регрессия Градиентный бустинг

Для заданного набора данных (по Вашему варианту) постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы 1 и 2 (по варианту для Вашей группы). Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик). Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей? Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д.

Импорт библиотек:

In [1]:

```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import balanced_accuracy_score, plot_roc_curve, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
```

In [4]:

```
# отбираем 5000 строк из всего датасета
data = pd.read_csv('data/hotel_bookings.csv', nrows=5000)
```

In [5]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 32 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   hotel                           5000 non-null   object
 1   is_canceled                     5000 non-null   int64
 2   lead_time                       5000 non-null   int64
 3   arrival_date_year               5000 non-null   int64
 4   arrival_date_month              5000 non-null   object
 5   arrival_date_week_number        5000 non-null   int64
 6   arrival_date_day_of_month       5000 non-null   int64
 7   stays_in_weekend_nights         5000 non-null   int64
 8   stays_in_week_nights            5000 non-null   int64
 9   adults                          5000 non-null   int64
 10  children                        5000 non-null   int64
 11  babies                          5000 non-null   int64
 12  meal                            5000 non-null   object
 13  country                         4998 non-null   object
 14  market_segment                  5000 non-null   object
 15  distribution_channel            5000 non-null   object
 16  is_repeated_guest               5000 non-null   int64
 17  previous_cancellations          5000 non-null   int64
 18  previous_bookings_not_canceled  5000 non-null   int64
 19  reserved_room_type              5000 non-null   object
 20  assigned_room_type              5000 non-null   object
 21  booking_changes                 5000 non-null   int64
 22  deposit_type                    5000 non-null   object
 23  agent                           4186 non-null   float64
 24  company                         292 non-null    float64
 25  days_in_waiting_list            5000 non-null   int64
 26  customer_type                   5000 non-null   object
 27  adr                             5000 non-null   float64
 28  required_car_parking_spaces     5000 non-null   int64
 29  total_of_special_requests       5000 non-null   int64
 30  reservation_status              5000 non-null   object
 31  reservation_status_date         5000 non-null   object
dtypes: float64(3), int64(17), object(12)
memory usage: 1.2+ MB
```

In [6]:

```
# Проверяем процент пропусков в данных для всех колонок
(data.isnull().sum()/data.shape[0]*100).sort_values(ascending=False)
```

Out[6]:

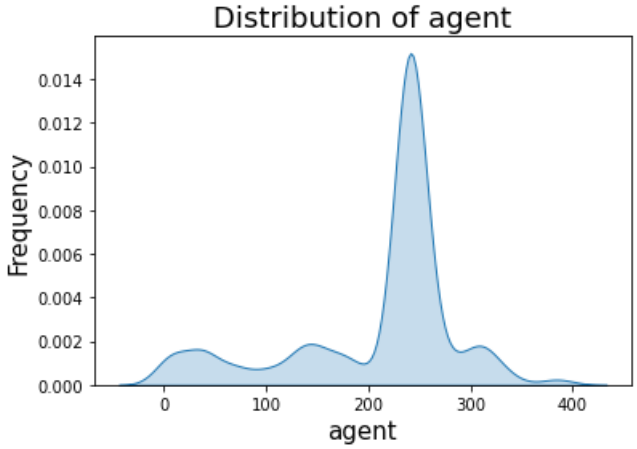
```
company                          94.16
agent                            16.28
country                           0.04
lead_time                         0.00
arrival_date_year                 0.00
arrival_date_month                0.00
arrival_date_week_number          0.00
is_canceled                       0.00
market_segment                    0.00
arrival_date_day_of_month         0.00
stays_in_weekend_nights           0.00
stays_in_week_nights              0.00
adults                            0.00
children                          0.00
babies                            0.00
meal                              0.00
reservation_status_date           0.00
distribution_channel              0.00
reservation_status                0.00
is_repeated_guest                 0.00
previous_cancellations            0.00
previous_bookings_not_canceled    0.00
reserved_room_type                0.00
assigned_room_type                0.00
booking_changes                   0.00
deposit_type                      0.00
days_in_waiting_list              0.00
customer_type                     0.00
adr                               0.00
required_car_parking_spaces       0.00
total_of_special_requests         0.00
hotel                             0.00
dtype: float64
```

In [7]:

```
# Строим гистограмму распределения для импутируемого признака
g = sns.kdeplot(data=data, x="agent", shade=True)
g.set_xlabel("agent", size = 15)
g.set_ylabel("Frequency", size = 15)
plt.title('Distribution of agent', size = 18)
```

Out[7]:

```
Text(0.5, 1.0, 'Distribution of agent')
```



Удалим строки, содержащие пропуски в столбце Country, так как оно имеет 0,04% пропусков;

Для пропущенных значений в столбце agent сделаем импутацию медианой, так как оно имеет 16,28% пропусков;

Удалим столбец Company, так как он имеет больше 90% пропусков

In [8]:

```
data.drop(['company'], axis=1, inplace=True)
```

In [9]:

```
data.dropna(subset=['country'], axis=0, inplace=True)
```

In [10]:

```
indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(data[['agent']])
imp_num = SimpleImputer(strategy='median')
data_num_imp = imp_num.fit_transform(data[['agent']])
```

```
data['agent'] = data_num_imp
filled_data = data_num_imp[mask_missing_values_only]
print('agent', 'median', filled_data.size, filled_data[0], filled_data[filled_data.size-1], sep='; ')
```

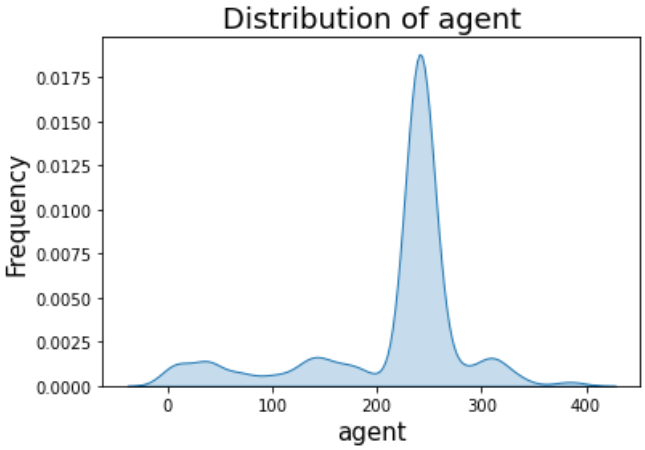
```
agent; median; 812; 240.0; 240.0
```

In [11]:

```
# Проверяем, что импутация не разрушила распределение
g = sns.kdeplot(data=data, x="agent", shade=True)
g.set_xlabel("agent", size = 15)
g.set_ylabel("Frequency", size = 15)
plt.title('Distribution of agent', size = 18)
```

Out[11]:

```
Text(0.5, 1.0, 'Distribution of agent')
```



In [12]:

```
# Копируем датасет и применяем label-encoding категориальных признаков для составления корреляционной матрицы
dataLE = data.copy()
le = LabelEncoder()
col_obj = dataLE.dtypes[dataLE.dtypes==object].index.values.tolist()
for i in col_obj:
    dataLE[i] = le.fit_transform(dataLE[i])
```

In [13]:

```
(dataLE.corr()['is_canceled']*100).sort_values(ascending=False)
```

Out[13]:

```
is_canceled                     100.000000
country                          52.533878
arrival_date_year                29.437152
deposit_type                     19.751308
lead_time                         7.588779
market_segment                    5.883349
distribution_channel              4.700574
adults                            4.537695
stays_in_weekend_nights           2.942242
children                          2.469151
stays_in_week_nights              0.049425
reservation_status_date          -0.040024
customer_type                    -0.979502
meal                             -1.987424
reserved_room_type               -2.664975
babies                           -2.954529
agent                            -3.553828
arrival_date_day_of_month        -3.558175
adr                              -4.973463
total_of_special_requests        -8.264548
days_in_waiting_list            -11.344538
arrival_date_month              -16.216285
booking_changes                 -18.118893
assigned_room_type              -19.255699
arrival_date_week_number        -24.489474
required_car_parking_spaces     -29.537194
reservation_status              -87.450209
hotel                                  NaN
is_repeated_guest                      NaN
previous_cancellations                 NaN
previous_bookings_not_canceled         NaN
Name: is_canceled, dtype: float64
```

In [21]:

```
del_data = (dataLE.corr()['is_canceled']*100).sort_values(ascending=False)
del_col = del_data[(del_data < 10) & (del_data > -10) | (del_data.isnull())].index.values.tolist()
data.drop(columns=del_col, inplace=True)
dataLE.drop(columns=del_col, inplace=True)
```

In [22]:

```
data.head()
```

Out[22]:

	is_canceled	arrival_date_year	arrival_date_week_number	booking_changes	days_in_waiting_list	required_car_parking_spaces	arrival_date_month_A
0	0	-0.580893	-0.318919	4.563780	-0.130859	-0.320296	
1	0	-0.580893	-0.318919	6.197718	-0.130859	-0.320296	
2	0	-0.580893	-0.318919	-0.338034	-0.130859	-0.320296	
3	0	-0.580893	-0.318919	-0.338034	-0.130859	-0.320296	
4	0	-0.580893	-0.318919	-0.338034	-0.130859	-0.320296	

5 rows × 82 columns

In [36]:

```
# Выполняем one-hot encoding и масштабирование
col_num = data.dtypes[data.dtypes!=object].index.values.tolist()
col_num.remove('is_canceled')
se = StandardScaler()
data[col_num] = se.fit_transform(data[col_num])
data = pd.get_dummies(data, drop_first=True)
```

In [37]:

```
dataLE_X = dataLE.drop(columns='is_canceled')
dataLE_y = dataLE['is_canceled']
data_X = data.drop(columns='is_canceled')
data_y = data['is_canceled']
```

In [39]:

```
dataLE_X_train, dataLE_X_test, dataLE_y_train, dataLE_y_test = train_test_split(dataLE_X, dataLE_y, \
                                                                                test_size = 0.3, \
                                                                                random_state= 1)
data_X_train, data_X_test, data_y_train, data_y_test = train_test_split(data_X, data_y, \
                                                                        test_size = 0.3, \
                                                                        random_state= 1)
```

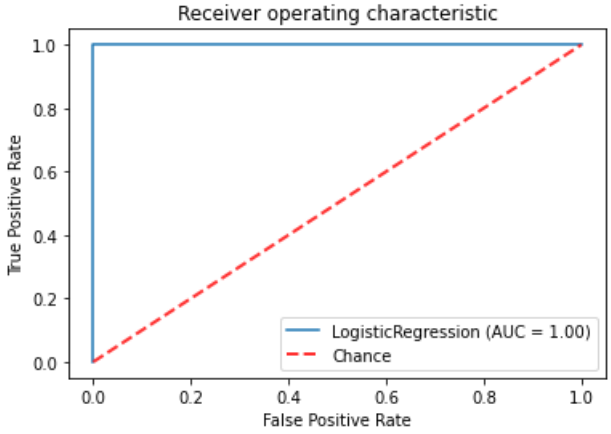
In [40]:

```
def print_metrics(X_train, Y_train, X_test, Y_test, clf):
    clf.fit(X_train, Y_train)
    target = clf.predict(X_test)
    print(f'Сбалансированная оценка: {balanced_accuracy_score(Y_test, target)}')
    fig, ax = plt.subplots()
    plot_roc_curve(clf, X_test, Y_test, ax=ax)
    ax.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
            label='Chance', alpha=.8)
    ax.set(xlim=[-0.05, 1.05], ylim=[-0.05, 1.05],
           title="Receiver operating characteristic")
    ax.legend(loc="lower right")
    plt.show()
    print(f'Матрица ошибок:\n {confusion_matrix(Y_test, target)}')
```

In [41]:

```
print_metrics(data_X_train, data_y_train, data_X_test, data_y_test, LogisticRegression())
```

```
Сбалансированная оценка: 1.0
```

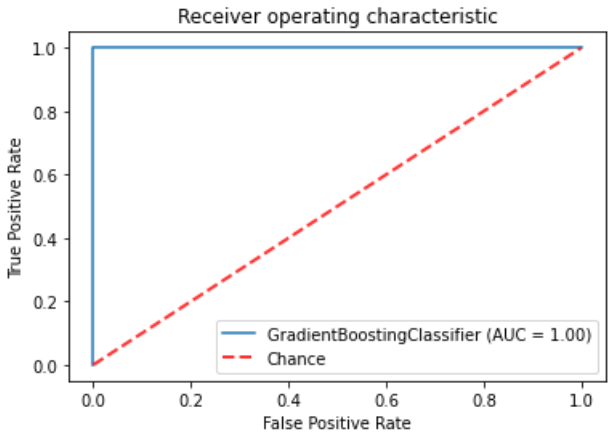



```
Матрица ошибок:
 [[810   0]
 [  0 690]]
```

In [42]:

```
print_metrics(dataLE_X_train, dataLE_y_train, dataLE_X_test, dataLE_y_test, GradientBoostingClassifier(random_
```

```
Сбалансированная оценка: 1.0
```




```
Матрица ошибок:
 [[810   0]
 [  0 690]]
```

Вывод

В качестве вывода об оценке качества моделей можно указать, что обе модели показали схожие результаты, а ошибки вызванные неверным определением некоторых классов могут быть вызваны неравномерностью распределения классов в датасете или при разбиении датасета на тренировочную и тестовую выборки

In [ ]: