Зудин Алексей Максимович, ИУ5-63Б Вариант №10: метод №1 -Дерево решений; метод №2 - Случайный лес.

Для рубежного контроля №2 согласно варианту взят [следующий датасет. (https://www.kaggle.com/jessemostipak/hotel-booking-demand)](https://www.kaggle.com/jessemostipak/hotel-booking-demand) Будем решать задачу бинарной классификации: будет ли отменено бронирование данной комнаты в отеле (**is_canceled** - целевой признак)

0. Подготовка

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
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
%matplotlib inline

from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

In [2]:
```
# отбираем 5000 строк из всего датасета
data = pd.read_csv('data/hotel_bookings.csv', nrows=5000)
```

In [3]:
```
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 [4]:
```
# Оцениваем баланс классов целевого признака
data['is_canceled'].value_counts()/data['is_canceled'].shape[0]*100
```

Out[4]:
```
0    54.04
1    45.96
Name: is_canceled, dtype: float64
```

In [5]:
```
# Проверяем процент пропусков в данных для всех колонок
(data.isnull().sum()/data.shape[0]*100).sort_values(ascending=False
```

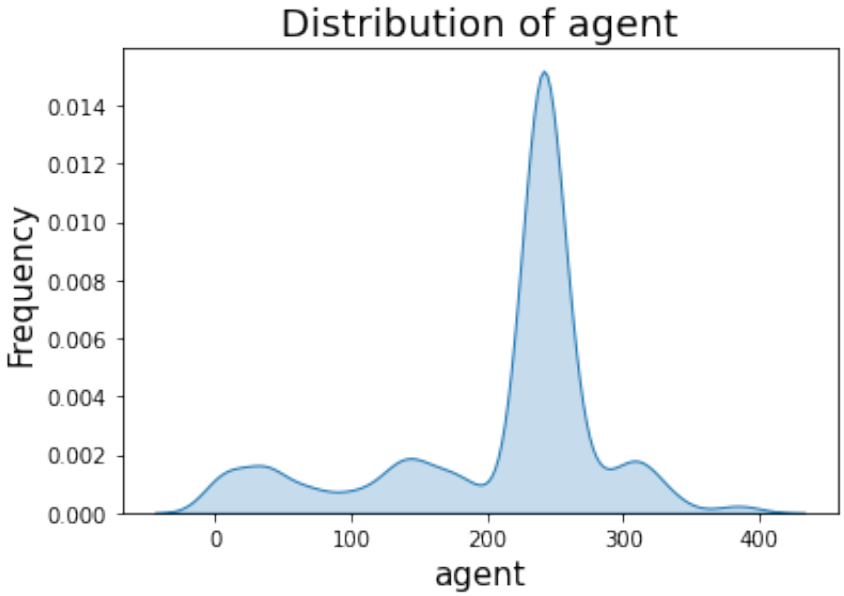
Out[5]:

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

dtype: float64

In [6]:
```
# 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[6]:
```
Text(0.5, 1.0, 'Distribution of agent')
```



Из анализа количества пропусков делаем следующие выводы:

- Строки, содержащие пропуски в столбце "country", удаляем;
- Для пропущенных значений в столбце "agent" сделаем импутацию медианой;
- Столбец "company" удаляем

In [7]:
```
data.drop(['company'], axis=1, inplace=True)
```

In [8]:
```
data.dropna(subset=['country'], axis=0, inplace=True)
```

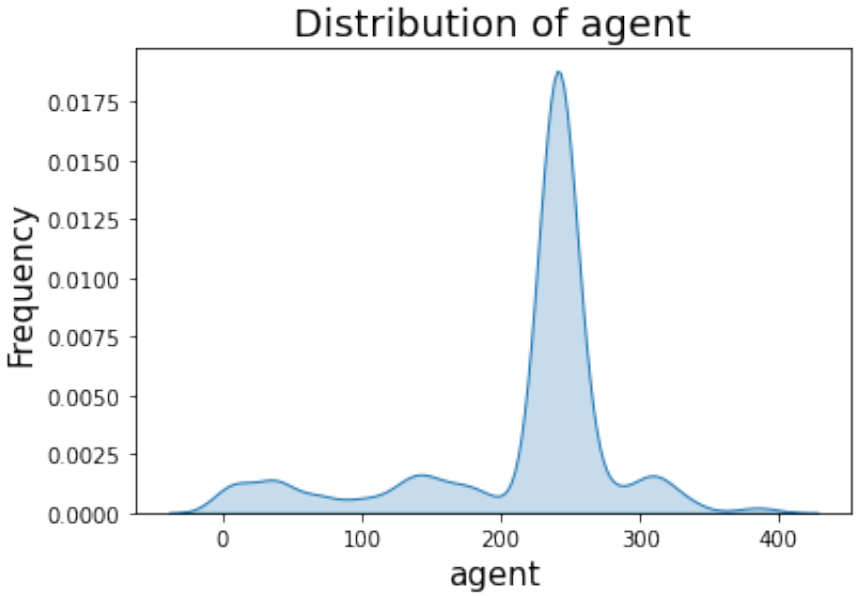
In [9]:
```
indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(data[['agent']])
imp_num = SimpleImputer(strategy='median')
data_num_imp = imp_num.fit_transform(data[['agent']])
data['agent'] = data_num_imp
filled_data = data_num_imp[mask_missing_values_only]
print('agent', 'median', filled_data.size, filled_data[0], filled_d
```

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

После применения импутации

In [10]:
```
# 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[10]:
```
Text(0.5, 1.0, 'Distribution of agent')
```



In [11]:
```
data.info()
```

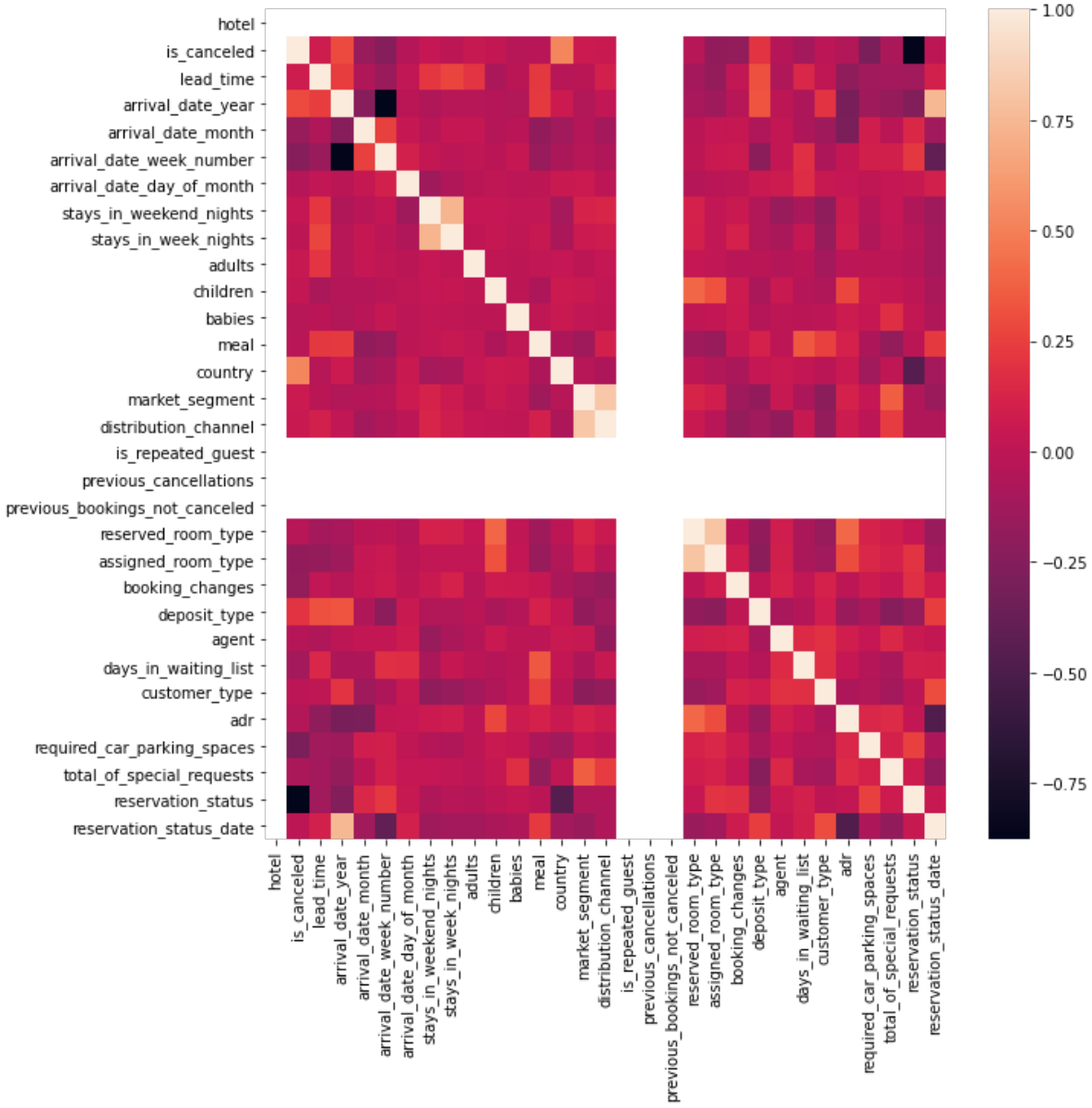
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4998 entries, 0 to 4999
Data columns (total 31 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   hotel                           4998 non-null   object 
 1   is_canceled                     4998 non-null   int64  
 2   lead_time                       4998 non-null   int64  
 3   arrival_date_year               4998 non-null   int64  
 4   arrival_date_month              4998 non-null   object 
 5   arrival_date_week_number        4998 non-null   int64  
 6   arrival_date_day_of_month       4998 non-null   int64  
 7   stays_in_weekend_nights         4998 non-null   int64  
 8   stays_in_week_nights            4998 non-null   int64  
 9   adults                          4998 non-null   int64  
 10  children                        4998 non-null   int64  
 11  babies                          4998 non-null   int64  
 12  meal                            4998 non-null   object 
 13  country                         4998 non-null   object 
 14  market_segment                  4998 non-null   object 
 15  distribution_channel            4998 non-null   object 
 16  is_repeated_guest               4998 non-null   int64  
 17  previous_cancellations          4998 non-null   int64  
 18  previous_bookings_not_canceled  4998 non-null   int64  
 19  reserved_room_type              4998 non-null   object 
 20  assigned_room_type              4998 non-null   object 
 21  booking_changes                 4998 non-null   int64  
 22  deposit_type                    4998 non-null   object 
 23  agent                           4998 non-null   float64
 24  days_in_waiting_list            4998 non-null   int64  
 25  customer_type                   4998 non-null   object 
 26  adr                             4998 non-null   float64
 27  required_car_parking_spaces     4998 non-null   int64  
 28  total_of_special_requests       4998 non-null   int64  
 29  reservation_status              4998 non-null   object 
 30  reservation_status_date         4998 non-null   object 
dtypes: float64(2), int64(17), object(12)
memory usage: 1.2+ MB
```

In [12]:
```
# Проверяем категориальные признаки на уникальность
col_obj = data.dtypes[data.dtypes==object].index.values.tolist()
for i in enumerate(col_obj):
    uniq_obj = data[i[1]].unique()
    print(f'{i[0]+1}. {i[1]}: {uniq_obj} | КОЛ-ВО: {len(uniq_obj)}'
```

```
 '2015-11-07' '2015-11-04' '2015-11-01' '2015-11-02' '2015-11-17'
 '2015-11-06' '2015-11-10' '2015-11-08' '2015-11-09' '2015-11-15'
 '2015-11-16' '2015-11-11' '2015-11-12' '2015-11-14' '2015-11-13'
 '2015-11-18' '2015-11-22' '2015-11-19' '2015-11-21' '2015-11-20'
 '2015-11-24' '2015-11-25' '2015-11-23' '2015-11-28' '2015-11-26'
 '2015-11-27' '2015-11-29' '2015-12-04' '2015-12-01' '2015-12-06'
 '2015-12-08' '2015-12-02' '2015-12-03' '2015-12-31' '2015-12-05'
 '2015-12-10' '2015-12-17' '2015-11-30' '2015-12-12' '2015-12-07'
 '2016-01-05' '2015-12-11' '2015-12-13' '2015-12-15' '2015-12-16'
 '2015-12-19' '2015-12-18' '2015-12-26' '2015-12-27' '2015-12-22'
 '2015-12-23' '2015-12-24' '2015-12-29' '2015-12-28' '2015-12-20'
 '2015-12-30' '2016-01-02' '2016-01-01' '2015-12-25' '2016-01-03'
 '2016-01-04' '2016-01-11' '2016-01-07' '2015-12-21' '2016-01-09'
 '2016-01-10' '2016-01-08' '2016-01-06' '2016-01-12' '2016-01-13'
 '2016-01-23' '2016-02-09' '2016-01-15' '2016-01-16' '2016-01-17'
 '2016-01-19' '2016-01-18' '2016-01-21' '2016-01-24' '2016-01-22'
 '2016-01-29' '2016-01-27' '2016-01-25' '2016-03-08' '2016-01-26'
 '2016-01-20' '2016-01-30' '2016-02-01' '2016-02-02' '2016-02-08'
 '2016-02-07' '2016-01-28' '2016-02-05' '2016-02-03' '2016-02-13'
```

In [13]:
```
# Копируем датасет и применяем label-encoding категориальных призна
# и последующего применения в модели Random Forest
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 [14]:
```
plt.figure(figsize=(10,10))
g = sns.heatmap(dataLE.corr())
```



In [15]:
```
# Оцениваем важность признаков для целевого
(dataLE.corr()['is_canceled']*100).sort_values(ascending=False)
```

Out[15]:
```
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
```

По результатам корреляционного анализа удаляем столбцы, которые имеют меньшую значимость по отношению к целевому признаку

In [16]:
```
del_data = (dataLE.corr()['is_canceled']*100).sort_values(ascending
del_col = del_data[(del_data < 10) & (del_data > -10) | (del_data.i
data.drop(columns=del_col, inplace=True)
dataLE.drop(columns=del_col, inplace=True)
```

In [17]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4998 entries, 0 to 4999
Data columns (total 11 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   is_canceled                  4998 non-null   int64 
 1   arrival_date_year            4998 non-null   int64 
 2   arrival_date_month           4998 non-null   object
 3   arrival_date_week_number     4998 non-null   int64 
 4   country                      4998 non-null   object
 5   assigned_room_type           4998 non-null   object
 6   booking_changes              4998 non-null   int64 
 7   deposit_type                 4998 non-null   object
 8   days_in_waiting_list         4998 non-null   int64 
 9   required_car_parking_spaces  4998 non-null   int64 
 10  reservation_status           4998 non-null   object
dtypes: int64(6), object(5)
memory usage: 468.6+ KB
```

Выполняем One-hot encoding для категориальных признаков и масштабирование числовых признаков для применения в SVM

In [18]:
```
# Выполняем one-hot encoding и масштабирование для применения в SVM
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 [19]:
```
TEST_SIZE = 0.3
RANDOM_STATE = 0
```

In [20]:
```
dataLE_X = dataLE.drop(columns='is_canceled')
dataLE_y = dataLE['is_canceled']
data_X = data.drop(columns='is_canceled')
data_y = data['is_canceled']
```

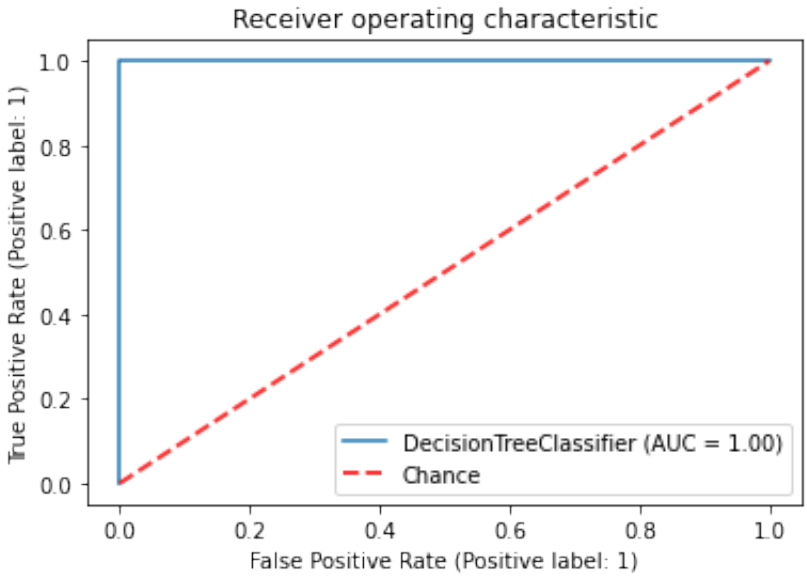
In [21]:
```
dataLE_X_train, dataLE_X_test, dataLE_y_train, dataLE_y_test = trai


data_X_train, data_X_test, data_y_train, data_y_test = train_test_s
```

In [22]:
```
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_tes
    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)}')
```

1. DecisionTreeClassifier

In [23]:
```
print_metrics(data_X_train, data_y_train, data_X_test, data_y_test,
```

```
Сбалансированная оценка: 1.0
```

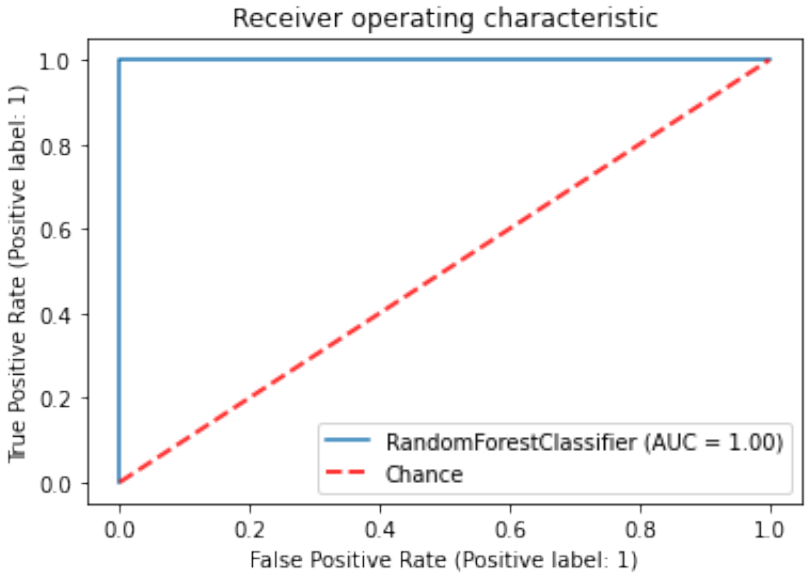



```
Матрица ошибок:
 [[827   0]
 [  0 673]]
```

2. Random Forest

In [24]:
```
print_metrics(dataLE_X_train, dataLE_y_train, dataLE_X_test, dataLE
```

```
Сбалансированная оценка: 1.0
```




```
Матрица ошибок:
 [[827   0]
 [  0 673]]
```

3. Выводы

В данной работе для оценки моделей были использованы следующие метрики, подходящие для задачи бинарной классификации:

- **balanced accuracy**, так как данная метрика хорошо интерпретируется и используется при несбалансированных классах
- **ROC-кривая (AUC)**, так как позволяет по графику понять, насколько модель может минимизировать FP (False Positive), т.е. признавать отмененным заказ, который таковым не является, и минимизировать FN (False Negative), т.е. признавать бронированным заказ, который был отменен
- **confusion matrix**, так как, хотя и метрикой в полной мере не является, позволяет увидеть общую картину по всем видам ошибок.

По результатам оценивания можно сделать следующий вывод: модель Random Forest обладает немного большей предсказательной способностью, чем Support Vector Machine. Но при этом обе модели могут использоваться для предсказания, будет ли заказ по бронированию отменен, с минимальным количеством ошибок.