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
```

```
---------------------------------------------------------------------
---------
ModuleNotFoundError                       Traceback (most recent c
all last)
<ipython-input-1-30a92b56e7eb> in <module>
      6 from sklearn.model_selection import train_test_split
      7 from sklearn.metrics import balanced_accuracy_score,
plot_roc_curve, confusion_matrix
----> 8 import seaborn as sns
      9 import matplotlib.pyplot as plt
     10 from sklearn.tree import DecisionTreeClassifier

ModuleNotFoundError: No module named 'seaborn'
```

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

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

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

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

In [ ]:
```
# 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)
```

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

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

In [ ]:
```
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
```

После применения импутации

In [ ]:
```
# 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)
```

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

In [ ]:
```
# 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)}'
```

In [ ]:
```
# 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 [ ]:
```
plt.figure(figsize=(10,10))
g = sns.heatmap(dataLE.corr())
```

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

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

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

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

In [ ]:
```
# 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 [ ]:
```
TEST_SIZE = 0.3
RANDOM_STATE = 0
```

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

2. Random Forest

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

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.