Датасет – Спрос на бронирование отелей:
https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand

**Обучение моделей и подбор параметров с использованием Grid Search**

В Grid Search используется параметр, отвечающий за кросс-валидацию, - cv.

Дерево решений. Автоматически с использованием Grid Search подбирается параметр: max_depth – глубина дерева. Она будет изменяться от 1 до 18 с шагом в 2. Лучшая модель имеет max_depth=13.

Ввод [21]:
```
from sklearn.model_selection import GridSearchCV

parametrs = {'max_depth': range (1,18, 2) }
dt = DecisionTreeClassifier()
grid = GridSearchCV(dt, parametrs, cv=3, n_jobs=-1)
grid.fit(x_train, y_train)
dt_best_params_ =  grid.best_params_
dt_best_params_
```

Out[21]:
```
{'max_depth': 3}
```

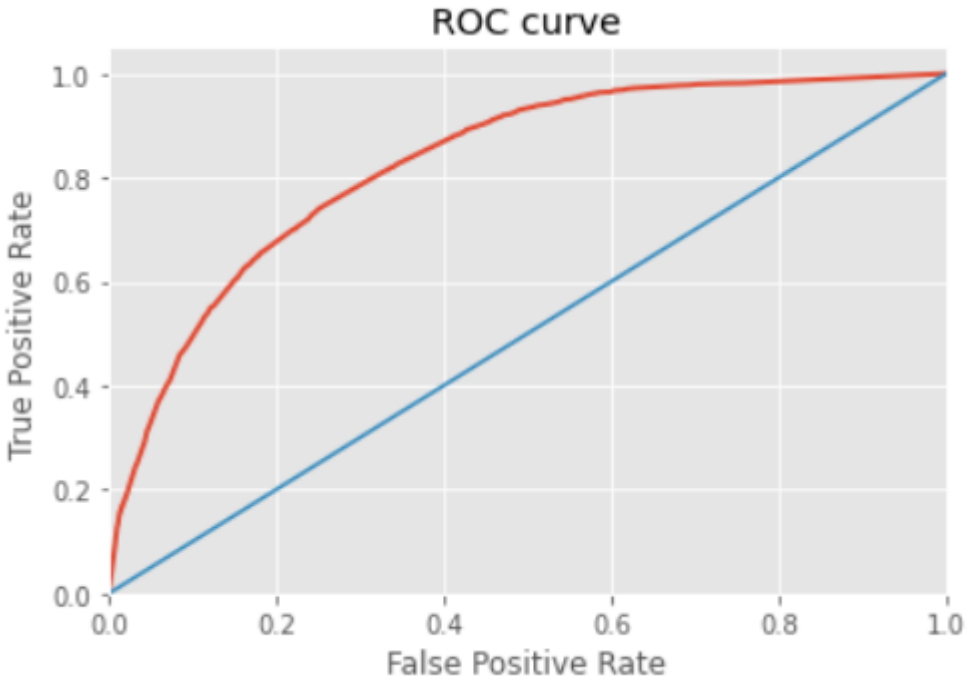
Ввод [22]:
```
from sklearn.tree import DecisionTreeClassifier
model_dt = DecisionTreeClassifier(**dt_best_params_)
model_dt.fit(x_train, y_train)
y_pred = model_dt.predict(x_test)

from sklearn import metrics
print(metrics.classification_report(y_test, y_pred))
```

Оценка качества прогноза (precision/recall/f1-score)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.83 | 0.88 | 0.86 | 11757 |
| 1 | 0.65 | 0.53 | 0.59 | 4657 |
| accuracy | | | 0.79 | 16414 |
| macro avg | 0.74 | 0.71 | 0.72 | 16414 |
| weighted avg | 0.78 | 0.79 | 0.78 | 16414 |

ROC-кривая




Случайный лес. Автоматически с использованием Grid Search подбираются следующие параметры: n_estimators – число деревьев в лесу. Оно будет изменяться от 10 до 110 с шагом 10, max_depth – глубина дерева. Она будет изменяться от 1 до 18 с шагом в 2. Лучшая модель имеет параметры n_estimators=60, max_depth=17.

Ввод [19]:
```
from sklearn.model_selection import GridSearchCV

parametrs = { 'n_estimators': range (10, 110, 10),
              'max_depth': range (1,18, 2) }
rfs = RandomForestClassifier()
grid = GridSearchCV(rfs, parametrs, cv=3)
grid.fit(x_train, y_train)
rfs_best_params_ =  grid.best_params_
rfs_best_params_
```

Out[19]:
```
{'max_depth': 13, 'n_estimators': 50}
```

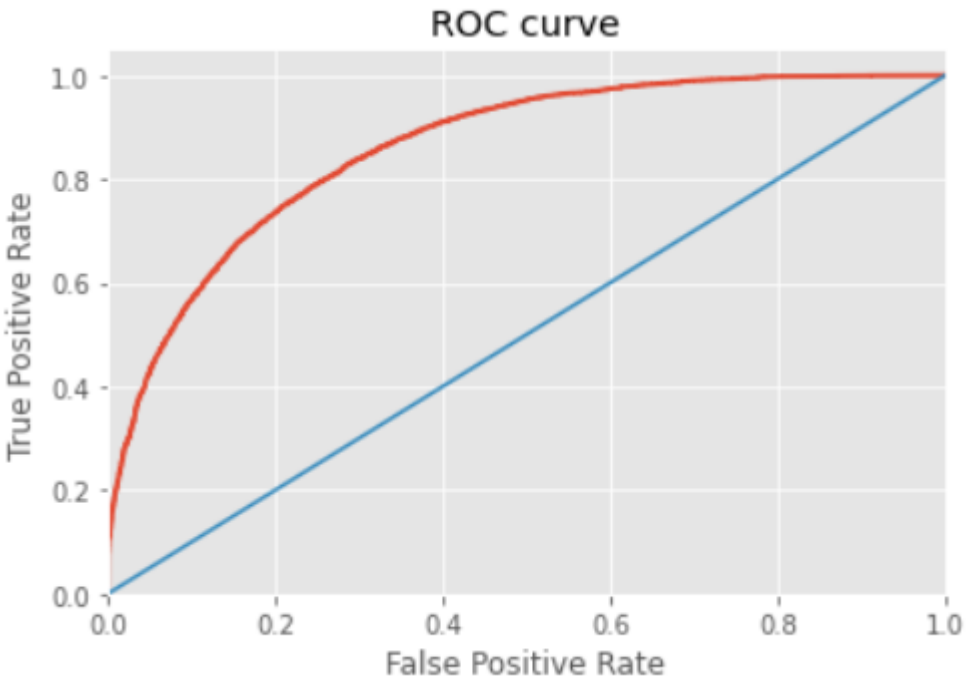
Ввод [20]:
```
from sklearn.ensemble import RandomForestClassifier
model_rf = RandomForestClassifier(**rfs_best_params_)
model_rf.fit(x_train, y_train)
y_pred = model_rf.predict(x_test)

from sklearn import metrics
print(metrics.classification_report(y_test, y_pred))
```

Оценка качества прогноза (precision/recall/f1-score)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.82 | 0.93 | 0.87 | 11757 |
| 1 | 0.74 | 0.48 | 0.59 | 4657 |
| accuracy | | | 0.80 | 16414 |
| macro avg | 0.78 | 0.71 | 0.73 | 16414 |
| weighted avg | 0.80 | 0.80 | 0.79 | 16414 |

ROC-кривая




KNN. Автоматически с использованием Grid Search подбираются следующие параметры: n_neighbors – n_ n_neighbors. Оно будет изменяться от 10 до 4, metric – метрика (L1, L2, Манхэттенское расстояние), weights – веса: «uniform»: все точки имеют одинаковый вес, «distance»: вес точек равен их расстоянию.

Ввод [*]:
```
from sklearn.model_selection import GridSearchCV

parametrs = {'n_neighbors': range (1,4),
             'metric': ['manhattan', 'l1', 'l2'],
             'weights': ['distance', 'uniform']}
knn = KNeighborsClassifier()
grid = GridSearchCV(knn, parametrs, cv=3, n_jobs=-1)
grid.fit(x_train, y_train)
knn_best_params_ =  grid.best_params_
knn_best_params_
```

Ввод [*]:
```
from sklearn.neighbors import KNeighborsClassifier
model_knn = KNeighborsClassifier(**knn_best_params_)
model_knn.fit(x_train, y_train)
y_pred = model_knn.predict(x_test)

from sklearn import metrics
print(metrics.classification_report(y_test, y_pred))
```

KNN требует не мало количества ресурсов, поэтому перебрать параметры на не самой быстрой машине не представляется возможным. Еще при условии, что датасет состоит из 80000+ строк и имеет 24 признака. Поэтому была построена только одна модель KNN с фиксированными признаками.

Ввод [11]:
```
from sklearn.neighbors import KNeighborsClassifier
model_knn = KNeighborsClassifier(n_neighbors=1,metric='manhattan',weights='distance')
model_knn.fit(x_train, y_train)
y_pred = model_knn.predict(x_test)

from sklearn import metrics
print(metrics.classification_report(y_test, y_pred))
```

Оценка качества прогноза (precision/recall/f1-score)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.80 | 0.77 | 0.79 | 11757 |
| 1 | 0.48 | 0.53 | 0.50 | 4657 |
| accuracy | | | 0.70 | 16414 |
| macro avg | 0.64 | 0.65 | 0.64 | 16414 |
| weighted avg | 0.71 | 0.70 | 0.71 | 16414 |

SVM. Требует много ресурсов для построения модели. Для этого способа не проводился подбор параметров с помощью GridSearchCV.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.73 | 0.98 | 0.84 | 11757 |
| 1 | 0.64 | 0.09 | 0.15 | 4657 |
| accuracy | | | 0.73 | 16414 |
| macro avg | 0.68 | 0.53 | 0.50 | 16414 |
| weighted avg | 0.70 | 0.73 | 0.64 | 16414 |

Визуализация деревьев решения

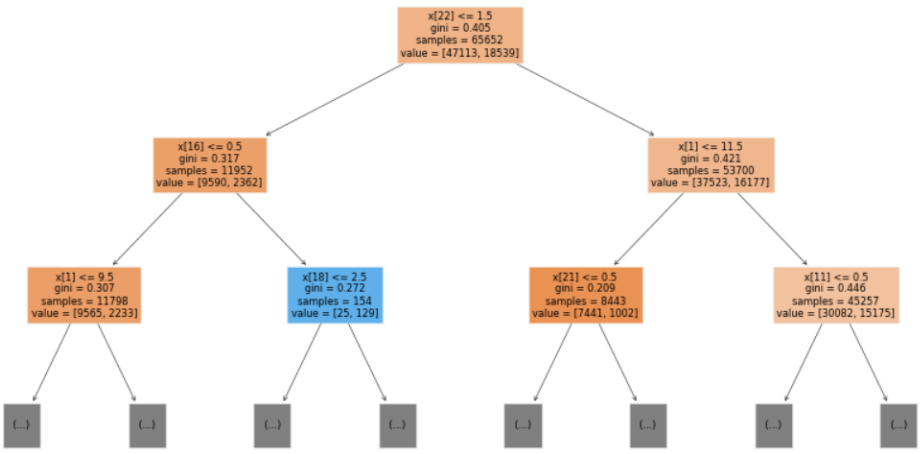
1) Первый способ – вывод текстом второй

Ввод [15]:
```
from sklearn import tree
text_representation = tree.export_text(model_dt)
print(text_representation)
```

```
|--- feature_22 <= 1.50
|   |--- feature_16 <= 0.50
|   |   |--- feature_1 <= 9.50
|   |   |   |--- feature_12 <= 0.50
|   |   |   |   |--- feature_14 <= 0.50
|   |   |   |   |   |--- feature_11 <= 0.50
|   |   |   |   |   |   |--- feature_0 <= 0.50
|   |   |   |   |   |   |   |--- feature_1 <= 5.50
|   |   |   |   |   |   |   |   |--- feature_8 <= 134.00
|   |   |   |   |   |   |   |   |   |--- feature_8 <= 82.50
|   |   |   |   |   |   |   |   |   |   |--- feature_8 <= 80.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 15
|   |   |   |   |   |   |   |   |   |   |--- feature_8 >  80.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
|   |   |   |   |   |   |   |   |   |--- feature_8 >  82.50
|   |   |   |   |   |   |   |   |   |   |--- feature_21 <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- feature_21 >  0.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
```

2) При помощи tree.plot_tree

Ввод [16]:
```
from sklearn import tree
plt.figure(figsize=(15, 8))
tree.plot_tree(model_dt,
               max_depth=2,
               filled = True);
plt.show()
```



Ввод [17]:

```
from sklearn import tree
plt.figure(figsize=(15, 8))
tree.plot_tree(model_dt,
               filled = True);
plt.show()
```