РК2 по дисциплине "Технологии машинного обучения"

Выполнила Попова Дарья, студентка РТ5-61Б

В качестве датасета для рубежного контроля будем использовать набор данных о мультфильмах компании Дисней. На основании имеющихся признаков нужно определить общий доход компании от проката каждого мультфильма.

Ссылка на датасет: https://www.kaggle.com/rashikrahmanpritom/disney-movies-19372016-total-gross

In [1]:

```
import pandas as pd
import numpy as np
disney = pd.read_csv('C:\\Users\\Дасупс\\Downloads\\disney_movies_total_gross.csv')
```

In [2]:

```
disney.head()
```

Out[2]:

	movie_title	release_date	genre	mpaa_rating	total_gross	inflation_adjusted_gross
0	Snow White and the Seven Dwarfs	1937-12-21	Musical	G	184925485	5228953251
1	Pinocchio	1940-02-09	Adventure	G	84300000	2188229052
2	Fantasia	1940-11-13	Musical	G	83320000	2187090808
3	Song of the South	1946-11-12	Adventure	G	65000000	1078510579
4	Cinderella	1950-02-15	Drama	G	85000000	920608730

In [3]:

```
disney.shape
```

Out[3]:

```
(579, 6)
```

In [4]:

```
disney.dtypes
```

Out[4]:

```
movie_title                object
release_date               object
genre                      object
mpaa_rating                object
total_gross                 int64
inflation_adjusted_gross    int64
dtype: object
```

Предобработка данных

Посмотрим, есть ли в данных пропуски.

In [5]:

```
disney.isnull().sum()
```

Out[5]:

```
movie_title                 0
release_date                0
genre                      17
mpaa_rating                56
total_gross                 0
inflation_adjusted_gross    0
dtype: int64
```

Пропуски есть в категориальных колонках genre и mpaa_rating. В обеих колонках пропуски составляют не более 10%, так что не будем исключать данные признаки из модели. Заполним пропуски.

In [6]:

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
disney['genre'] = imputer.fit_transform(disney[['genre']])
disney['mpaa_rating'] = imputer.fit_transform(disney[['mpaa_rating']])
```

In [7]:

```
disney.isnull().any()
```

Out[7]:

```
movie_title                False
release_date               False
genre                      False
mpaa_rating                False
total_gross                False
inflation_adjusted_gross   False
dtype: bool
```

In [8]:

```
disney['total_gross'].corr(disney['inflation_adjusted_gross'])
```

Out[8]:

```
0.4270642887713366
```

Сделаем предположение, что название фильма и дата его выхода никак не влияет на его успех, и удалим соответствующую колонку movie_title.

In [9]:

```
disney = disney.drop(['release_date','movie_title'], axis=1)
```

Закодируем категориальные признаки с жанром и возрастным ограничением (рейтингом) мультфильма.

In [10]:

```
disney = pd.get_dummies(disney)
```

In [11]:

```
disney.head()
```

Out[11]:

	total_gross	inflation_adjusted_gross	genre_Action	genre_Adventure	genre_Black Comedy	genre_Comedy	genre_Concert/Performance	genre_Document
0	184925485	5228953251	0	0	0	0	0	
1	84300000	2188229052	0	1	0	0	0	
2	83320000	2187090808	0	0	0	0	0	
3	65000000	1078510579	0	1	0	0	0	
4	85000000	920608730	0	0	0	0	0	

Разделение данных

In [12]:

```
X = disney.drop(['total_gross'], axis=1)
y = disney['total_gross']
```

In [13]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

Обучение моделей

Заранее определим функцию для вывода метрик. Для оценки качества работы моделей будем использовать классические регрессионные метрики:

- коэффициент детерминации R^2
- среднюю абсолютную ошибку MAE
- среднюю квадратичную ошибку MSE

In [14]:

```
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

def print_metrics(y_true, y_predicted):
    print("{:<15} {:<15}".format('метрика','значение'))
    print()
    print("{:<15} {:<15}".format('r2_score', round(r2_score(y_true, y_predicted),3)))
    print("{:<15} {:<15}".format('MAE', round(mean_absolute_error(y_true, y_predicted),1)))
    print("{:<15} {:<15}".format('MSE', round(mean_squared_error(y_true, y_predicted),1)))
```

Решающее дерево

In [15]:

```
from sklearn.tree import DecisionTreeRegressor
tree_cl = DecisionTreeRegressor(max_depth=5, random_state=42)
```

```
tree_cl.fit(X_train, y_train)
tree_cl_predicted = tree_cl.predict(X_test)
print_metrics(y_test, tree_cl_predicted)
```

```
метрика         значение

r2_score        0.764
MAE             18617948.6
MSE             1179326967057425.8
```

Градиентный бустинг

In [16]:

```
from sklearn.ensemble import GradientBoostingRegressor

# default n_estimators=100

grad_boost = GradientBoostingRegressor(random_state=42)
grad_boost.fit(X_train, y_train)
boost_predicted = grad_boost.predict(X_test)

print_metrics(y_test, boost_predicted)
```

```
метрика         значение

r2_score        0.728
MAE             18201217.1
MSE             1358824262894962.8
```

Итоги

Итак, обе модели (и единичная - "слабая" - с решающим деревом, и ансамблевая - градиентный бустинг) показывают примерно одни и те же результаты качества работы (довольно средненькие).

In [ ]: