РК №2 ТМО

Казакова В.В. ИУ5Ц-81Б

Вариант №32

Тема: Методы построения моделей машинного обучения

Методы: Линейная/логичтическая регрессия, Случайный лес

Датасет: https://www.kaggle.com/rashikrahmanpritom/disney-movies-19372016-total-gross

Датасет представляет собой набор данных о фильмах Disney с 1937 по 2016 годы, и содержит следующую информацию:\ – название фильма\ – дата выпуска\ – жанр\ – рейтинг MPAA\ – общая касса (total_gross)\ – касса с учетом инфляции (inflation_adjusted_gross)

На основе имеющих данных, выберем задачу регрессии для прогнозирования кассовых сборов с учетом инфляции (inflation_adjusted_gross), так как это непрерывная целевая переменная

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.impute import SimpleImputer

# Загрузка данных
data = pd.read_csv('disney_movies_total_gross.csv')

# Предобработка данных
# Преобразование даты в числовые признаки
data['release_date'] = pd.to_datetime(data['release_date'])
data['release_year'] = data['release_date'].dt.year
data['release_month'] = data['release_date'].dt.month
data['release_day'] = data['release_date'].dt.day

# Заполнение пропусков в рейтинге MPAA
data['mpaa_rating'] = data['mpaa_rating'].fillna('Unknown')

# Кодирование категориальных признаков
label_encoder = LabelEncoder()
data['genre_encoded'] = label_encoder.fit_transform(data['genre'].fillna('Unknown'))
data['mpaa_encoded'] = label_encoder.fit_transform(data['mpaa_rating'])

# Удаление строк с отсутствующими целевыми значениями
data = data[data['inflation_adjusted_gross'] > 0]

# Выбор признаков и целевой переменной
features = ['release_year', 'release_month', 'release_day', 'genre_encoded', 'mpaa_encoded']
X = data[features]
y = data['inflation_adjusted_gross']

# Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Построение моделей

1. Линейная регрессия

In [2]:
```
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)

# Оценка модели
mse_lr = mean_squared_error(y_test, y_pred_lr)
r2_lr = r2_score(y_test, y_pred_lr)

print(f"Linear Regression - MSE: {mse_lr:.2f}, R2: {r2_lr:.2f}")
```

```
Linear Regression - MSE: 41972586629100488.00, R2: 0.28
```

2. Случайный лес

In [5]:
```
from sklearn.ensemble import RandomForestRegressor
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2', None]
}

rf = RandomForestRegressor(random_state=42)

grid_search = GridSearchCV(
    estimator=rf,
    param_grid=param_grid,
    cv=5,
    scoring='neg_mean_squared_error',
    n_jobs=-1,
    verbose=2
)

# Обучение с подбором параметров
grid_search.fit(X_train, y_train)

print("Лучшие параметры:", grid_search.best_params_)

# Предсказание и оценка
best_rf = grid_search.best_estimator_
y_pred_rf = best_rf.predict(X_test)

mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

print(f"Optimized Random Forest - MSE: {mse_rf:.2f}, R2: {r2_rf:.2f}")
```

```
Fitting 5 folds for each of 324 candidates, totalling 1620 fits
Лучшие параметры: {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_est
imators': 50}
Optimized Random Forest - MSE: 19617749533072084.00, R2: 0.66
```

Метрики качества и интерпретация

Для оценки моделей регрессии были использованы:
Mean Squared Error (MSE) — средняя квадратичная ошибка, которая показывает среднее квадратичное отклонение предсказаний от истинных значений. Чем меньше MSE, тем лучше.
R-squared (R^2) — коэффициент детерминации, который показывает долю дисперсии зависимой переменной, объясняемую моделью. Диапазон значений: от 0 до 1, где 1 означает идеальное соответствие модели данным.

Результаты
Линейная регрессия: MSE ~4.20×10^{16} и R^2 ~0.28
Случайный лес: MSE**: ~2.13×10^{16} и R^2 ~0.63

Выводы 1) Качество моделей 1.Случайный лес показал значительно лучшее качество, чем линейная регрессия:

```
  - R² = 0.63 против 0.28, что означает, что случайный лес объясняет 63% дисперсии кассовых
сборов, тогда как линейная модель — только 28%
  - MSE случайного леса почти в 2 раза ниже, что подтверждает его более точные предсказания
```

1. Линейная регрессия всё ещё недостаточно эффективна, что говорит о том, что линейные зависимости слабо описывают влияние признаков на кассовые сборы
2. Случайный лес демонстрирует хорошее качество ($R^2 = 0.63$), что можно считать приемлемым для сложной задачи прогнозирования кассовых сборов

2) Возможные улучшения

1. Для случайного леса:

- Оптимизация гиперпараметров (глубина деревьев, количество деревьев, критерий разбиения).
- Ансамблирование (например, переход на Gradient Boosting).
- Учет взаимодействий признаков (feature engineering).

1. Для линейной регрессии:

- Попробовать регуляризацию.
- Логарифмирование целевой переменной, если распределение сборов имеет долгий хвост.

3) Общие улучшения для обеих моделей:

- Добавление новых признаков:
  - Бюджет фильма
  - Продолжительность
  - Актерский состав (наличие топ-актеров)
  - Временные факторы (сезон премьеры, конкуренция с другими релизами)
  - Данные из внешних источников: рейтинги IMDb, маркетинговый бюджет, отзывы критиков.

4) Интерпретация

- Случайный лес дает осмысленные предсказания, но это можно улучшить.
- Линейная регрессия в текущем виде малопригодна для задачи.
- Основная проблема — кассовые сборы зависят от множества факторов, не учтенных в данных (маркетинг, реклама, зрительские ожидания, конкуренция).
- Для практического применения нужно больше данных и более сложные модели (например, ансамбли).