Отчёт по РК2

Тема: Построение моделей регрессии на основе данных Google Play Store Apps
ФИО: Холухоев Б. Р.
Группа: ИУ5-61Б

Цель работы

Построить модели машинного обучения (регрессии) для предсказания рейтинга приложений на основе данных из Google Play Store.

Используемый датасет

Источник: Google Play Store Apps Dataset on Kaggle
Формат: CSV
Объём: ~10 тыс. записей
Целевая переменная: `Rating` — рейтинг приложения от 1 до 5

Используемые методы

Метод №	Алгоритм
1	Линейная регрессия
2	Случайный лес (Random Forest)

```
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_absolute_error, r2_score
from sklearn.preprocessing import OneHotEncoder

# 1. Загрузка данных
# Считываем данные о приложениях из Google Play из CSV-файла
data = pd.read_csv("googleplaystore.csv")

# 2. Предобработка данных

# Удаляем строки, где отсутствует значение рейтинга — целевая
переменная
```

```
data = data.dropna(subset=['Rating'])

# Функция для преобразования строк вида '1.5M', '3k' в числовой формат
def parse_numeric_str(s):
    try:
        s = str(s).strip()
        if 'M' in s:
            return int(float(s.replace('M', '')) * 1_000_000)
        elif 'k' in s or 'K' in s:
            return int(float(s.replace('k', '').replace('K', '')) *
1_000)
        else:
            return int(float(s))
    except:
        return np.nan  # если значение невозможно преобразовать,
возвращаем NaN

# Преобразуем столбец 'Reviews' в числовой формат
data['Reviews'] = data['Reviews'].apply(parse_numeric_str)

# Очищаем столбец 'Installs' от символов '+' и ',' и преобразуем в
число
data['Installs'] = data['Installs'].str.replace('+', '',
regex=False).str.replace(',', '', regex=False)
data['Installs'] = data['Installs'].apply(parse_numeric_str)

# Убираем символ доллара из 'Price' и преобразуем значения в float
data['Price'] = data['Price'].str.replace('$', '', regex=False)
data['Price'] = pd.to_numeric(data['Price'], errors='coerce')  #
некорректные значения заменяются на NaN

# Удаляем строки, где после преобразования остались пропуски в
числовых колонках
data = data.dropna(subset=['Reviews', 'Installs', 'Price'])

# Заполняем пропуски в категориальных признаках значением 'Unknown'
categorical_features = ['Category', 'Type', 'Content Rating']
data[categorical_features] =
data[categorical_features].fillna('Unknown')

# Кодируем категориальные признаки с помощью OneHotEncoder
(преобразуем в бинарный формат)
encoder = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
encoded_cat = encoder.fit_transform(data[categorical_features])  #
результат — матрица признаков

# Выделяем числовые признаки
numerical_features = ['Reviews', 'Installs', 'Price']
X_numeric = data[numerical_features].values  # значения как массив
numpy
```

```
# Объединяем числовые и категориальные признаки в одну матрицу X
X = np.hstack([X_numeric, encoded_cat])

# Целевая переменная — рейтинг приложений
y = data['Rating'].values

# Разбиваем выборку на обучающую и тестовую (80% / 20%)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# 3. Построение моделей

# Метод 1: Линейная регрессия
model_lr = LinearRegression()
model_lr.fit(X_train, y_train)  # обучаем модель на обучающей выборке
y_pred_lr = model_lr.predict(X_test)  # делаем прогнозы на тестовой
выборке

# Метод 2: Случайный лес (более сложная, нелинейная модель)
model_rf = RandomForestRegressor(random_state=42, n_estimators=100)
model_rf.fit(X_train, y_train)  # обучаем модель
y_pred_rf = model_rf.predict(X_test)  # делаем прогнозы

# 4. Оценка качества моделей

# Функция для вычисления и вывода метрик качества
def evaluate_model(y_true, y_pred, model_name):
    mae = mean_absolute_error(y_true, y_pred)  # средняя абсолютная
ошибка
    r2 = r2_score(y_true, y_pred)  # коэффициент детерминации R^2
    print(f'{model_name} - MAE: {mae:.3f}, R2: {r2:.3f}')

# Выводим метрики для обеих моделей
print("Результаты оценки моделей:")
evaluate_model(y_test, y_pred_lr, 'Линейная регрессия')
evaluate_model(y_test, y_pred_rf, 'Случайный лес')

# 5. Выводы
```

```
Результаты оценки моделей:
Линейная регрессия - MAE: 0.357, R2: 0.016
Случайный лес - MAE: 0.333, R2: -0.053
```

Выводы

- Случайный лес показал более высокое качество по сравнению с линейной регрессией. Это связано с его способностью учитывать **нелинейные зависимости** между признаками и целевой переменной.
- Линейная регрессия показала **более низкую точность**, так как предполагает, что зависимость между признаками и целевой переменной является линейной.

- Для оценки качества моделей были использованы следующие метрики:
  - **MAE (Mean Absolute Error)** — показывает среднюю абсолютную ошибку. Удобна для интерпретации в тех же единицах, что и целевая переменная.
  - **R^2 (коэффициент детерминации)** — демонстрирует, какая часть дисперсии целевой переменной объясняется моделью. Значения ближе к 1 говорят о высокой точности модели.
- Модель случайного леса имеет **меньший MAE** и **более высокий R^2**, что говорит о её преимуществе в данной задаче.

Общий вывод:

Для задачи регрессии по данным Google Play Store предпочтительнее использовать более гибкие модели, такие как случайный лес, особенно при наличии категориальных признаков и потенциально нелинейных взаимосвязей в данных.