

Курзанова Анастасия ИУ5-61Б 8 вариант

Задание. Для заданного набора данных (по Вашему варианту) постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы 1 и 2 (по варианту для Вашей группы). Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик). Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей? Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д.

ИУ5-61Б, ИУ5Ц-81Б Линейная/логистическая регрессия Случайный лес

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import accuracy_score, confusion_matrix
```

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

```
# Выведем первые 5 строк датасета
data.head()
```

↗

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design	January 7, 2018	1.0.0	4.0.3 and up
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play	January 15, 2018	2.0.0	4.0.3 and up
2	U Launcher Lite – FREE Live Cool Themes	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	August 1, 2018	1.2.4	4.0.3 and up

◀ ▶

Далее: ☒ [Посмотреть рекомендованные графики](#)

```
# Размер
data.shape
```

↗ (10841, 13)

```
# Список колонок с типами данных
data.dtypes
```

↗

App	object
Category	object
Rating	float64
Reviews	object
Size	object
Installs	object
Type	object
Price	object
Content Rating	object
Genres	object
Last Updated	object
Current Ver	object
Android Ver	object
dtype:	object

```
# Проверим наличие пустых значений
for col in data.columns:
```

```
# Количество пустых значений - все значения заполнены
temp_null_count = data[data[col].isnull()].shape[0]
print('{} - {}'.format(col, temp_null_count))
```

```
App - 0
Category - 0
Rating - 1474
Reviews - 0
Size - 0
Installs - 0
Type - 1
Price - 0
Content Rating - 1
Genres - 0
Last Updated - 0
Current Ver - 8
Android Ver - 3
```

```
# Удаление строк, содержащих пустые значения
df = data.dropna(axis=0, how='any')
(data.shape, df.shape)
```

```
((10841, 13), (9360, 13))
```

```
for col in df.columns:
    # Количество пустых значений - все значения заполнены
    temp_null_count = df[df[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))
total_count = df.shape[0]
print('Всего строк: {}'.format(total_count))
```

```
App - 0
Category - 0
Rating - 0
Reviews - 0
Size - 0
Installs - 0
Type - 0
Price - 0
Content Rating - 0
Genres - 0
Last Updated - 0
Current Ver - 0
Android Ver - 0
Всего строк: 9360
```

```
if 'Size' in df.columns:
    df.drop('Size', axis=1, inplace=True)
else:
    print("'Size' column does not exist in the DataFrame.")
```

```
for column in df.columns:
    # Для категориальных столбцов используем наиболее часто встречающееся значение
    if df[column].dtype == 'object':
        df[column] = df[column].fillna(df[column].mode()[0])
    else:
        # Для числовых столбцов можно использовать медиану
        df[column] = df[column].fillna(df[column].median())
```

```
df = df[df['Installs']!= 'Free']
```

```
# Преобразуем все значения в столбце 'Installs' в строки
df['Installs'] = df['Installs'].apply(lambda x: str(x))
```

```
# Теперь можем безопасно использовать.str для преобразования формата 'Installs' к числовому
df['Installs'] = df['Installs'].str.replace('+', '').str.replace(',', '').astype(int)
```

```
# Преобразуем все значения в столбце 'Price' в строки
df['Price'] = df['Price'].apply(lambda x: str(x))
```


```
# Теперь можем безопасно использовать.str для преобразования формата 'Price' к числовому
df['Price'] = df['Price'].str.replace('$', '').astype(float)
```

```
# Кодирование категориальных признаков
label_encoder = LabelEncoder()
categorical_columns = ['Category', 'Type', 'Content Rating', 'Genres', "Last Updated", "Current Ver", "Android Ver"]
```

```
for col in categorical_columns:
    df[col] = label_encoder.fit_transform(df[col])
```

```
# Преобразуем 'Reviews' в числовой формат
df['Reviews'] = df['Reviews'].astype(int)
```

```
# Целевая переменная и признаки
X = df.drop(['App', 'Rating'], axis=1)
y = df['Rating']
```

 <ipython-input-75-7001d1cc8a5a>:2: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus)  
df.drop('Size', axis=1, inplace=True)  
<ipython-input-75-7001d1cc8a5a>:9: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus)  
df[column] = df[column].fillna(df[column].mode()[0])  
<ipython-input-75-7001d1cc8a5a>:12: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus)  
df[column] = df[column].fillna(df[column].median())  
<ipython-input-75-7001d1cc8a5a>:9: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus)  
df[column] = df[column].fillna(df[column].mode()[0])


```
# Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
# Модели
lin_reg = LinearRegression()
```

```
# Обучение моделей
lin_reg.fit(X_train, y_train)
```

```
# Предсказания моделей
y_pred_lin = lin_reg.predict(X_test)
```

```
# Метрики качества моделей
lin_r2 = r2_score(y_test, y_pred_lin)
lin_mse = mean_squared_error(y_test, y_pred_lin)
mae_rf = mean_absolute_error(y_test, y_pred_lin)
print("Linear Regression:")
print(f'Linear Regression R^2: {lin_r2}')
print(f'Linear Regression MSE: {lin_mse}')
print(f'Linear Regression MAE: {mae_rf}')
```

 Linear Regression:  
Linear Regression R^2: 0.013879145005462279  
Linear Regression MSE: 0.26753674086718726  
Linear Regression MAE: 0.35497447529767073

```
# Обучение модели
rf_regressor = RandomForestRegressor(random_state=42)
rf_regressor.fit(X_train, y_train)
```

```
# Предсказания
y_pred_rf = rf_regressor.predict(X_test)
```

```
# Оценка качества модели
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)
```

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

# Обучение модели
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train, y_train)

# Предсказания
y_pred_rf = rf_model.predict(X_test)

# Оценка качества модели
mse_rf = mean_squared_error(y_test, y_pred_rf)
mae_rf = mean_absolute_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

print("Random Forest Regressor:")
print("Mean Squared Error:", mse_rf)
print("Mean Absolute Error:", mae_rf)

print("R^2 Score:", r2_rf)
```

```
➞ Random Forest Regressor:
Mean Squared Error: 0.21990145571175104
Mean Absolute Error: 0.30156042048229537
R^2 Score: 0.18945932129499354
```

Для оценки качества моделей будем использовать следующие метрики:

Среднеквадратичная ошибка (MSE): Отражает среднее квадратичное отклонение предсказаний от фактических значений. Чем меньше MSE, тем лучше.

R-квадрат ( $R^2$ ): Определяет долю дисперсии зависимой переменной, объясняемую моделью.  $R^2$  ближе к 1 указывает на лучшую модель.

Mean Absolute Error (MAE) - это метрика, которая показывает среднее абсолютное отклонение прогнозируемых значений модели от фактических значений на тестовой выборке. При сравнении двух моделей по MAE нам нужно сравнить их значения MAE: чем меньше MAE, тем лучше модель.

Linear Regression:

Linear Regression  $R^2$ : 0.013879145005462279

Linear Regression MSE: 0.26753674086718726

Linear Regression MAE: 0.35497447529767073

Random Forest Regressor:

Mean Squared Error: 0.21990145571175104

Mean Absolute Error: 0.30156042048229537

$R^2$  Score: 0.18945932129499354

**По результатам оценки качества видно, что Random Forest Regressor превосходит линейную регрессию по трем метрикам**

Не удается связаться с сервисом reCAPTCHA. Проверьте подключение к Интернету и перезагрузите страницу.