Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»
Отчет по рубежному контролю №2
«Методы построения моделей машинного обучения»
Вариант №8

Выполнил:
студент группы ИУ5-61Б
Коваленко Алексей
Викторович

Подпись:_____

Дата:_____

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий
Евгеньевич

Подпись:_____

Дата:_____

Москва, 2023 г.

Выполнение работы

Для выполнения задачи построения моделей классификации был представлен набор данных Google Play Store Apps

Ввод [1]:
```
import time
import datetime
import re
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import  mean_squared_error, mean_absolute_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import  RandomForestRegressor
from sklearn import preprocessing
```

Ввод [3]:
```
dataset = pd.read_csv('./googleplaystore.csv')
dataset.head()
```

Out[3]:

	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, Hide ...	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
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	Art & Design	June 8, 2018	Varies with device	4.2 and up
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity	June 20, 2018	1.1	4.4 and up

Наш набор данных содержит 1 целевой признак и 12 нецелевых.

In [5]:
```
wine_ds = pd.DataFrame(data=wine['data'], columns=wine['feature_names'])
wine_ds['target'] = wine['target']
wine_ds
```

Out[5]:

alinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline	target
15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065.0	0
11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050.0	0
18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185.0	0
16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480.0	0
21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735.0	0
...	...	...	...	...	...	...	...	...	...	...
20.5	95.0	1.68	0.61	0.52	1.06	7.70	0.64	1.74	740.0	2
23.0	102.0	1.80	0.75	0.43	1.41	7.30	0.70	1.56	750.0	2
20.0	120.0	1.59	0.69	0.43	1.35	10.20	0.59	1.56	835.0	2
20.0	120.0	1.65	0.68	0.53	1.46	9.30	0.60	1.62	840.0	2
24.5	96.0	2.05	0.76	0.56	1.35	9.20	0.61	1.60	560.0	2

Типы данных всех полей являются различными, поэтому необходимо сначала их преобразовать.

Ввод [4]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10841 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   App             10841 non-null  object
 1   Category        10841 non-null  object
 2   Rating          9367 non-null   float64
 3   Reviews         10841 non-null  object
 4   Size            10841 non-null  object
 5   Installs        10841 non-null  object
 6   Type            10840 non-null  object
 7   Price           10841 non-null  object
 8   Content Rating  10840 non-null  object
 9   Genres          10841 non-null  object
 10  Last Updated    10841 non-null  object
 11  Current Ver     10833 non-null  object
 12  Android Ver     10838 non-null  object
dtypes: float64(1), object(12)
memory usage: 1.1+ MB
```

Но перед этим обработаем пропуски.

Ввод [6]:
```
dataset.isnull().sum()
```

Out[6]:
```
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
dtype: int64
```

Для этого удалим строки с пустыми значениями

Ввод [7]:
```
dataset.dropna(subset=['Content Rating', 'Rating','Current Ver', 'Android Ver'], axis=0, inplace=True)
dataset.describe(include="all").T
```

Out[7]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
App	9360	8190	ROBLOX	9	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Category	9360	33	FAMILY	1746	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Rating	9360.0	NaN	NaN	NaN	4.191838	0.515263	1.0	4.0	4.3	4.5	5.0
Reviews	9360	5990	2	83	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Size	9360	413	Varies with device	1637	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Installs	9360	19	1,000,000+	1576	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Type	9360	2	Free	8715	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Price	9360	73	0	8715	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Content Rating	9360	6	Everyone	7414	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Genres	9360	115	Tools	732	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Last Updated	9360	1299	August 3, 2018	319	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Current Ver	9360	2638	Varies with device	1415	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Android Ver	9360	31	4.1 and up	2059	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Начнем преобразование данных. Для цены уберем символ “$” из строки и приведем к float. Для жанров классифицируем их через LabelEncoder. Для Загрузок удалим все лишние символы. Для даты последнего обновления приведем значение к ко времени из строки. Отзывы приведем просто к целому числовому значению. А для размера преобразуем значение, удалив буквы и приведя числа к одному формату.

Ввод [9]:
```
le = preprocessing.LabelEncoder()
dataset['Price'] = dataset['Price'].apply(lambda x : float(x.strip('$')))
dataset['Genres'] = le.fit_transform(dataset['Genres'])
dataset['Installs'] = dataset['Installs'].apply(lambda x : int(x.strip('+').replace(',', '')))
dataset['Last Updated'] = dataset['Last Updated'].apply(lambda x : time.mktime(datetime.datetime.strptime(x, '%B %d, %Y').timetu
dataset['Reviews'] = dataset['Reviews'].apply(lambda x : int(x))
dataset['Size'] = dataset['Size'].apply(lambda x: float(x.strip('M')) if x.find('M')!=-1 else round(float(x.strip('k'))/1024,3)
dataset.head()
```

Out[9]:

	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	19.0	10000	Free	0.0	Everyone	3	1.515272e+09	1.0.0	4.0.3 and up
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14.0	500000	Free	0.0	Everyone	3	1.515964e+09	2.0.0	4.0.3 and up
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7	5000000	Free	0.0	Everyone	3	1.533071e+09	1.2.4	4.0.3 and up
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25.0	50000000	Free	0.0	Teen	3	1.528405e+09	Varies with device	4.2 and up
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8	100000	Free	0.0	Everyone	3	1.529442e+09	1.1	4.4 and up

Ввод [10]:
```
dataset.info()
```

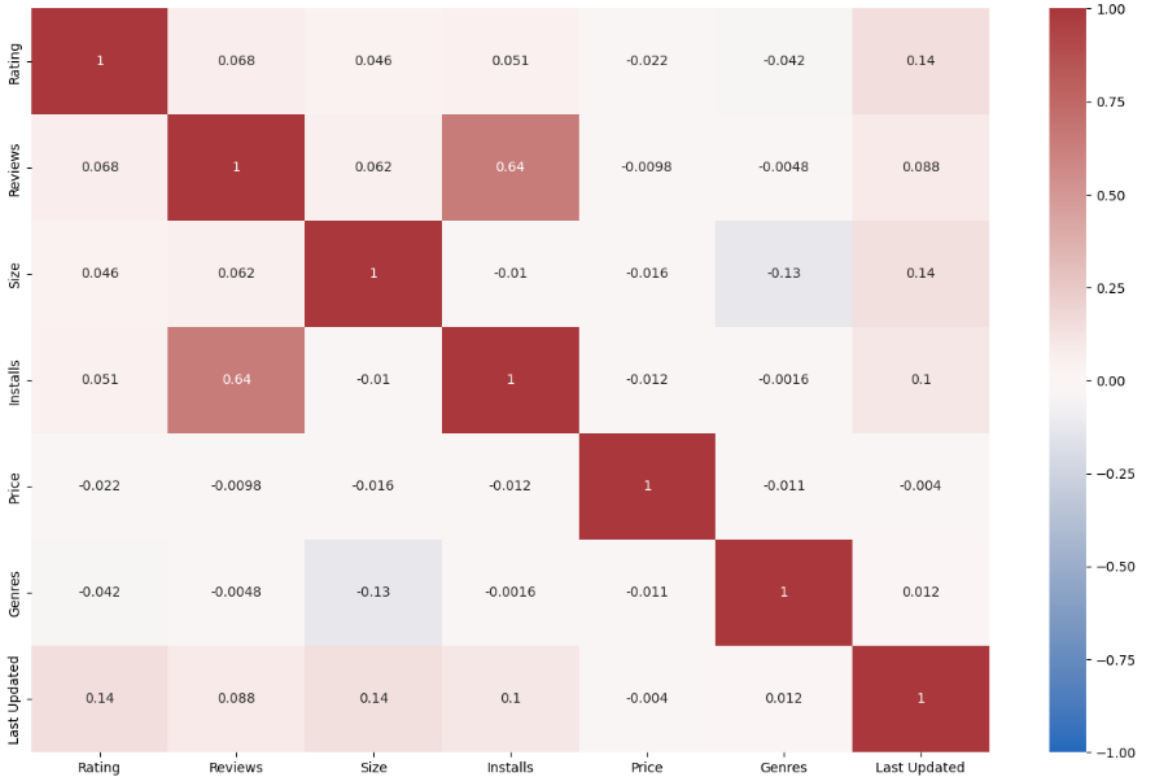
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9360 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   App             9360 non-null   object 
 1   Category        9360 non-null   object 
 2   Rating          9360 non-null   float64
 3   Reviews         9360 non-null   int64  
 4   Size            9360 non-null   float64
 5   Installs        9360 non-null   int64  
 6   Type            9360 non-null   object 
 7   Price           9360 non-null   float64
 8   Content Rating  9360 non-null   object 
 9   Genres          9360 non-null   int32  
 10  Last Updated    9360 non-null   float64
 11  Current Ver     9360 non-null   object 
 12  Android Ver     9360 non-null   object 
dtypes: float64(4), int32(1), int64(2), object(6)
memory usage: 987.2+ KB
```

Проведем корреляционный анализ на основе преобразованной части данных, чтобы оценить вклад признаков для построения моделей классификации. Для визуализации корреляционной матрицы была использована “тепловая карта”.

```
Ввод [11]: plt.figure(figsize=(16,10))
           sns.heatmap(dataset.corr(),annot=True, vmin=-1.0, vmax = 1.0,cmap = "vlag")
```

```
C:\Users\Necron\AppData\Local\Temp\ipykernel_22888\3537643971.py:2: FutureWarning: The default value of numeric_only in DataFra
me.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric
_only to silence this warning.
  sns.heatmap(dataset.corr(),annot=True, vmin=-1.0, vmax = 1.0,cmap = "vlag")
```

```
Out[11]: <Axes: >
```



По итогам данного анализа можно сказать, что нет особо коррелирующих между собой признаков.

Продолжим преобразования, разделив несколько столбцов и удалив из них лишние.

Rating Reviews Size Installs Price Genres Last Updated

Ввод [12]:
```
replaces = [u'\u00AE', u'\u2013', u'\u00C3', u'\u00E3', u'\u00B3', '[', ']', "'"]
for i in replaces:
    dataset['Current Ver'] = dataset['Current Ver'].astype(str).apply(lambda x : x.replace(i, ''))

regex = [r'[-+|/:/;(_)@]', r'\s+', r'[A-Za-z]+']
for j in regex:
    dataset['Current Ver'] = dataset['Current Ver'].astype(str).apply(lambda x : re.sub(j, '0', x))

dataset['Current Ver'] = dataset['Current Ver'].astype(str).apply(lambda x : x.replace('.', ',',1).replace('.', '').replace(',',
dataset['Current Ver'] = dataset['Current Ver'].fillna(dataset['Current Ver'].median())
```

Ввод [13]:
```
dummy_C = pd.get_dummies(dataset["Category"])
dataset = pd.concat([dataset, dummy_C], axis = 1)
dummy_G = pd.get_dummies(dataset["Type"])
dataset = pd.concat([dataset, dummy_G], axis = 1)
dummy_CR = pd.get_dummies(dataset['Content Rating'])
dataset = pd.concat([dataset, dummy_CR], axis = 1)
dataset.drop(['App', 'Android Ver',  'Category',"Type",'Content Rating' ], axis=1, inplace=True)
dataset.head()
```

Out[13]:

	Rating	Reviews	Size	Installs	Price	Genres	Last Updated	Current Ver	ART_AND_DESIGN	AUTO_AND_VEHICLES	...	VIDEO_PLAYERS	WEATHER	Free	P
0	4.1	159	19.0	10000	0.0	3	1.515272e+09	1.00	1	0	...	0	0	1	
1	3.9	967	14.0	500000	0.0	3	1.515964e+09	2.00	1	0	...	0	0	1	
2	4.7	87510	8.7	5000000	0.0	3	1.533071e+09	1.24	1	0	...	0	0	1	
3	4.5	215644	25.0	50000000	0.0	3	1.528405e+09	0.00	1	0	...	0	0	1	
4	4.3	967	2.8	100000	0.0	3	1.529442e+09	1.10	1	0	...	0	0	1	

5 rows × 49 columns

Далее на основе полученных признаков сделаем выборку для создания модели, которая будет включать в себя признаки: “Category”, “Rating”, “Reviews”, “Size”, “Installs”, “Type”, “Price”, “Content Rating”, “Genres”, “Last Updated” и “Current ver”.

Ввод [14]:
```
x  = dataset.iloc[:, 1:].values
y = dataset.iloc[:,0].values
x
```

Out[14]:
```
array([[1.59000e+02, 1.90000e+01, 1.00000e+04, ..., 0.00000e+00,
        0.00000e+00, 0.00000e+00],
       [9.67000e+02, 1.40000e+01, 5.00000e+05, ..., 0.00000e+00,
        0.00000e+00, 0.00000e+00],
       [8.75100e+04, 8.70000e+00, 5.00000e+06, ..., 0.00000e+00,
        0.00000e+00, 0.00000e+00],
       ...,
       [4.00000e+00, 3.60000e+00, 1.00000e+02, ..., 0.00000e+00,
        0.00000e+00, 0.00000e+00],
       [1.14000e+02, 0.00000e+00, 1.00000e+03, ..., 1.00000e+00,
        0.00000e+00, 0.00000e+00],
       [3.98307e+05, 1.90000e+01, 1.00000e+07, ..., 0.00000e+00,
        0.00000e+00, 0.00000e+00]])
```

Ввод [15]:
```
y
```

Out[15]:
```
array([4.1, 3.9, 4.7, ..., 5. , 4.5, 4.5])
```

Затем разделим выборки на тестовую и обучающую.

Ввод [16]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2,shuffle=True)
```

Для оценки качества модели будем использовать коэффициент детерминации, среднюю абсолютную ошибку, среднеквадратичное отклонение и среднеквадратичную ошибку.

```
from sklearn import preprocessing
```

Ввод [2]:
```
def ResultsShow(y_train,y_test, y_predict_train,  y_predict_test):
    print('TRAIN SET:')
    print ('  Mean Squared Error: '+ str(mean_squared_error(y_train,y_predict_train)))
    print ('  Root Mean Squared Error: '+ str(mean_squared_error(y_train,y_predict_train, squared=False)))
    print ('  Mean absolute Error: '+ str(mean_absolute_error(y_train,y_predict_train)))
    print ('  R2_score: '+ str(r2_score(y_train,y_predict_train)))
    print('TEST SET:')
    print ('  Mean Squared Error: '+ str(mean_squared_error(y_test,y_predict_test)))
    print ('  Root Mean Squared Error: '+ str(mean_squared_error(y_train,y_predict_train, squared=False)))
    print ('  Mean absolute Error: '+ str(mean_absolute_error(y_test,y_predict_test)))
    print ('  R2_score: '+ str(r2_score(y_test,y_predict_test)))
```

Сделаем модель линейной регрессии.

Ввод [17]:
```
linear = LinearRegression()
linear.fit(x_train,y_train)
y_pred_test_linear = linear.predict(x_test)
y_pred_train_linear = linear.predict(x_train)
ResultsShow(y_train,y_test,y_pred_train_linear, y_pred_test_linear)
```

```
TRAIN SET:
  Mean Squared Error: 0.26189201061041417
  Root Mean Squared Error: 0.5117538574455636
  Mean absolute Error: 0.358747491378218
  R2_score: 4.713189807470375e-05
TEST SET:
  Mean Squared Error: 0.279805073320546
  Root Mean Squared Error: 0.5117538574455636
  Mean absolute Error: 0.363354287031995
  R2_score: -0.0015395760105074707
```

Сделаем модель случайного леса

Ввод [18]:
```
tree = RandomForestRegressor(n_estimators=200, random_state=10)
tree.fit(x_train, y_train)
y_pred_test_tree = tree.predict(x_test)
y_pred_train_tree = tree.predict(x_train)
ResultsShow(y_train,y_test,y_pred_train_tree, y_pred_test_tree)
```

```
TRAIN SET:
  Mean Squared Error: 0.029234010569077127
  Root Mean Squared Error: 0.17097956184607893
  Mean absolute Error: 0.1099104300213674
  R2_score: 0.8883790588092563
TEST SET:
  Mean Squared Error: 0.22074271145733174
  Root Mean Squared Error: 0.17097956184607893
  Mean absolute Error: 0.2971170272435897
  R2_score: 0.2098693600629873
```

Таким образом, можно сказать, что обе модели имеет довольно сомнительное качество предсказания результатов. Я считаю, что данная проблема вызвана тем, что правильно предсказать оценку приложения

довольно сложно основываясь только на этих данных, так как на неё влияют множество факторов, которые не могут быть отражены в приложении.