Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Отчет по РК № 2

Технологии машинного обучения

Студент:

группы ИУ5-64Б

Ведьгун Е.А.

2021 г.

Москва

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.

Группа: ИУ5-64Б

Метод №1: Линейная/логистическая регрессия

Метод №2: Градиентный бустинг

Импорт библиотек

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRegressor
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_error, median_absolute_error, r2_score
%matplotlib inline
sns.set(style="ticks")
```

Загрузка датасета. Разделение выборки на тестовую и обучающую

```
bc = load_breast_cancer()
```

```
# Сформируем DataFrame
bc_df = pd.DataFrame(data= np.c_[bc['data']],
                     columns= bc['feature_names'])
bc_df
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area	w smoothr
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	25.380	17.33	184.60	2019.0	0.16
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	24.990	23.41	158.80	1956.0	0.12
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	23.570	25.53	152.50	1709.0	0.14
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	14.910	26.50	98.87	567.7	0.20
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	22.540	16.67	152.20	1575.0	0.13
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	25.450	26.40	166.10	2027.0	0.14
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	23.690	38.25	155.00	1731.0	0.11
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	18.980	34.12	126.70	1124.0	0.11
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	25.740	39.42	184.60	1821.0	0.16
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	9.456	30.37	59.16	268.6	0.08

569 rows × 30 columns

```
sc = MinMaxScaler()
bc_sc = sc.fit_transform(bc_df)
X_train, X_test, Y_train, Y_test = train_test_split(
bc_sc, bc.target, test_size=0.33, random_state=1)
```

Обучение и тестирование линейной регрессии

```
reg = LinearRegression().fit(X_train, Y_train.reshape(-1, 1))
reg.coef_, reg.intercept_
```

```
(array([[ 6.07255942,  0.22447383, -5.01865477, -0.43155119, -0.18415226,
          0.82846448, -0.23061375, -0.21391254, -0.00731892,  0.33764182,
         -0.33621402,  0.21432846,  0.12158385, -0.01268547, -0.72524925,
          0.29091039,  1.23928723, -0.53274651, -0.3783309 , -0.06403036,
         -5.53412912, -0.63584833,  0.85068656,  3.65615183,  0.27324166,
          0.13475022, -0.71811631, -0.4201767 , -0.17077405, -0.76773468]]),
 array([1.66248298]))
```

```
target = reg.predict(X_test)
mean_squared_error(Y_test, target), mean_absolute_error(Y_test, target)
```

```
(0.06092697018150137, 0.19403126210377075)
```

Обучение и тестирование градиентного бустинга

```
gr_boost_bc = GradientBoostingRegressor(random_state=1)
gr_boost_bc.fit(X_train, Y_train)
target2 = gr_boost_bc.predict(X_test)
mean_squared_error(Y_test, target2), mean_absolute_error(Y_test, target2)
```

```
(0.039501125391911175, 0.09537203204500203)
```

Выводы

Высокая эффективность градиентного бустинга в данном примере обуславливается тем, что выбранный датасет содержит много сложных зависимостей, а линейная регрессия отработала хуже, потому что это модель не ансамблиевая, а также датасет не содержит большое количество линейный зависимостей.