Московский государственный технический университет имени Н. Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5

Отчёт по
рубежному контролю № 2
«Технологии машинного обучения»

Подготовил:

Кан Андрей Дмитриевич

Группа ИУ5-64Б

Подпись_____

Дата_____

Москва
2021г.

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.

- **Для студентов групп ИУ5-61Б, ИУ5-62Б, ИУ5-63Б, ИУ5-64Б, ИУ5-65Б, РТ5-61Б номер варианта = номер в списке группы.**
- **Для студентов групп ИУ5Ц-81Б, ИУ5Ц-82Б, ИУ5Ц-83Б номер варианта = 25 + номер в списке группы.**
- При решении задач можно выбирать любое подмножество признаков из приведенного набора данных.
- Для сокращения времени построения моделей можно использовать фрагмент набора данных (например, первые 200-500 строк).
- Методы 1 и 2 для каждой группы приведены в следующей таблице:

Группа	Метод №1	Метод №2
ИУ5-61Б, ИУ5Ц-81Б	Линейная/логистическая регрессия	Случайный лес
ИУ5-62Б, ИУ5Ц-82Б	Метод опорных векторов	Случайный лес
ИУ5-63Б, ИУ5Ц-83Б	Дерево решений	Случайный лес
ИУ5-64Б	Линейная/логистическая регрессия	Градиентный бустинг
ИУ5-65Б	Метод опорных векторов	Градиентный бустинг
РТ5-61Б	Дерево решений	Градиентный бустинг

Выполнение:

import numpy as np

```
import pandas as pd
from sklearn.datasets import *
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt

from operator import itemgetter
import matplotlib.ticker as ticker
import math

from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score,
classification_report
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_absolute_error, mean_squared_error,
mean_squared_log_error, median_absolute_error, r2_score
from sklearn.metrics import roc_curve, roc_auc_score


from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRegressor
from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.model_selection import KFold, RepeatedKFold, LeaveOneOut, LeavePOut,
ShuffleSplit, StratifiedKFold



from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer

from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.model_selection import learning_curve, validation_curve

from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import SGDRegressor
from sklearn.linear_model import SGDClassifier
from typing import Dict, Tuple
from scipy import stats
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor,
export_graphviz

%matplotlib inline
sns.set(style="ticks")
```

Выборка датасета и ее разделение на тестовую и обучающую

```
wine = load_wine()

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

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_tensit
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_tensit
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32
...	...	...	...	...	...	...	...	...	...	...
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	0.52	1.06	7.70
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	0.43	1.41	7.30
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	0.43	1.35	10.20
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	0.53	1.46	9.30
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	0.56	1.35	9.20

178 rows × 13 columns

```
sc = MinMaxScaler()
wine_sc = sc.fit_transform(wine_df)
X_train, X_test, Y_train, Y_test = train_test_split(
wine_sc, wine.target, test_size=0.33, random_state=1)
```

Обучение и тестирование моделей

Обучение и тестирование линейной регрессии

```
reg1 = LinearRegression().fit(X_train, Y_train.reshape(-1, 1))
reg1.coef_, reg1.intercept_
```

```
(array([[-0.3976376 ,  0.06900004, -0.09455592,  0.67398288, -0.28642767,
          0.52588975, -1.74765428, -0.17922208,  0.03652941,  0.96993586,
         -0.36623749, -0.65402594, -1.20252002]]),
 array([1.9579414]))
```

```
target1 = reg1.predict(X_test)
mean_squared_error(Y_test, target1), mean_absolute_error(Y_test, target1)
```

```
(0.07434617175982262, 0.21742003782587782)
```

Обучение и тестирование градиентного бустинга

```
# Важность признаков
gr_boost_wine = GradientBoostingRegressor(random_state=1)
gr_boost_wine.fit(X_train, Y_train)
target2 = gr_boost_wine.predict(X_test)
mean_squared_error(Y_test, target2), mean_absolute_error(Y_test, target2)
```

```
(0.03364781195186767, 0.06616976908385173)
```

Выводы

Судя по метрикам, градиентный бустинг отработал лучше линейной регрессии. Высокая эффективность градиентного бустинга в данном примере обуславливается тем, что выбранный датасет содержит много сложных зависимостей, а линейная регрессия отработала хуже, во-первых, потому что это обычная модель, а не ансамблиевая, во вторых, потому что датасет не содержит большое количество линейный зависимостей.